End-User Driven Business Process Composition

vom Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

Dissertation

zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)

von

Diplom-Ingenieur

Todor Stoitsev

aus Sofia, Bulgarien

Referent: Prof. Dr. Max Mühlhäuser
Korreferent: Dr. Fabio Paternò

Tag der Einreichung: 05 Mai 2009
Tag der Mündlichen Prüfung: 17 Juni 2009

Darmstadt 2009
Hochschulkennziffer D17

Ehrenwörtliche Erklärung

Hiermit erkläre ich, die vorgelegte Arbeit zur Erlangung des akademischen Grades “Dr.-Ing.” mit dem Titel “End-User Driven Business Process Composition” selbständig und ausschließlich unter Verwendung der angegebenen Hilfsmittel erstellt zu haben. Ich habe bisher noch keinen Promotionsversuch unternommen.

Darmstadt, den 30.04.2009 Dipl.-Ing. Todor Stoitsev

Acknowledgments

Writing a dissertation is an ambitious endeavor which is hardly possible without support from various directions. This dissertation is not an exception. Therefore I would like to thank all people who supported me on my way to the doctoral degree.

My thanks go to:

- My advisor Prof. Dr. Max Mühlhäuser for providing me with valuable guidance and teaching me to look beyond the horizon. This dissertation would not have been possible without him.
- My co-supervisor Dr. Fabio Paternò for the kind support and constructive feedback on the concepts developed in the thesis.
- Dr. Knut Manske who believed in me and encouraged all my achievements on the path to the dissertation.
- Dr. Stefan Scheidl for his extraordinary engagement in helping me to settle down in Germany, for preparing a fruitful ground for the research in the thesis and for assisting me in the different phases of my research.
- Dr. Felix Flentge for the constructive feedback on concepts developed in the thesis.
- Dr. Nicolay Mehanjiev for the fruitful cooperation on the empirical questionnaire studies and for providing guidance on end-user development topics.
- Dr. Uwe Riss and Olaf Grebner for the cooperation on the task management and task pattern concepts.
- Michael Spahn for the teamwork and for being with me in the trenches.
- Dr. Birgit Zimmermann for the constructive feedback on the research concepts and for taking care of the research project and leaving me space to work on the dissertation.
- Victoria Carlsson for her vital support in the preliminary empirical studies and task analysis.
- Daniel Zwicker, Teena Vellaramkalayil, Axel Schulz, Simon Stebbins, and Markus Wiemann for their contributions to the development of the CTM system.
- All colleagues from the Telecooperation Group at the Darmstadt University of Technology and at SAP Research CEC Darmstadt for the friendly working atmosphere and the valuable feedback on my research in the doctoral seminars.
- All people from the industrial partner companies who participated in the user studies and evaluation phases.
- Above all, I would like to thank to my beloved wife Brigita Zareva-Stoitseva, for her love and care, for helping me not to forget the really important things in life and for making it all worthwhile.

Darmstadt, April 2009

Abstract

Business Process Management (BPM) solutions enable enterprises to consolidate and optimize their business operations and to gain competitive advantage in the fast evolving global market. Often, the only ones to understand the matter and complexity of business processes are the end users of enterprise software, who execute them on a daily basis. The need to involve end users in business process composition during the implementation of BPM solutions in enterprises is clearly perceived. However, end users have a detailed domain expertise but limited technical skills. Therefore upfront process modeling through conventional modeling notations remains inaccessible for them. The need for user-centric process composition approaches arises, which can enable end users to tailor business processes according to their actual expertise and problem solving strategies. Furthermore, these approaches need to bridge the process understanding of end users and technically skilled process designers and developers in the course of workflow projects in order to facilitate the development of real-life compliant and consistent process models and to streamline the uptake of BPM software in enterprises.

This thesis addresses end-user driven composition of both: (i) weakly-structured process models for supporting underspecified, human-centric business processes and (ii) structured business process models for automation of rigidly recurring processes through workflow engines. Both process types are composed through *programming by example* in a collaborative task management system. Task management is chosen as a starting point for end-user driven process composition in order to reconcile the personal and the enterprise perspectives on business processes. Programming by example is an *end-user development* technique, which enables capturing and repeated execution of user activities in a software system. The application of this technique in an enterprise scope for the composition of business process models is novel and requires specific support from user's perspective and from formal system's perspective.

The four major *scientific contributions* of the thesis can be captured as: (i) a task management model for human-centric business processes; (ii) a method for composition of weakly-structured process models through collaborative task management; (iii) a method for transformation of weakly-structured process models to structured workflows and their refinement based on deviations with ad-hoc tasks at runtime; (iv) the holistic concept for end-user driven business process composition through programming by example, composing contributions (i) through (iii) into a seamless overarching method and architecture for the composition of weakly-structured and structured process models. The elaborated concepts provide a significant contribution to known process modeling approaches in various research areas such as human-computer interaction, BPM, workflow management and computer supported cooperative work.

The presented concepts found on preliminary empirical studies, comprising an online questionnaire distributed to a number of companies from various industries, and a series of field studies in three German small and medium enterprises. The preliminary studies deliver strong support for end-user development in the domain of task management and identify entry points for introducing process tailoring to end users. These studies provide input for the elaborated task management model and drive the design choices for the architecture, underling the presented holistic concept.

The presented concepts take into consideration existing end user work practices and software applications for management of day-to-day activities, such as email and personal to-do list applications delivered with standard office environments. The task management model enables aggregation of data from these applications for the composition of weakly-structured business process models. These models can be repeatedly adapted and reused for the execution of ad-hoc, human-centric processes. The method for generation of structured workflows from weakly-structured process models enables automation of rigidly recurring processes through workflow engines. Generated workflows can be extended by process designers and developers, in a shared

context between user-defined and formal process models. The mapping of weakly-structured process models to structured workflow models facilitates data reuse between ad-hoc and operational processes. Enhanced data sharing and interoperability between ad-hoc and structured processes is enabled through the introduced holistic concept and the underlying architecture.

The presented concepts have been implemented and validated through a prototype called Collaborative Task Manager. The evaluation results confirm that the proposed end-user development approach and its enterprise-wide application through the presented concepts efficiently enable end-user driven business process composition. Thus the specified task management model, methods and holistic concept can be used for designing user-tailorable BPM systems that facilitate the adoption of BPM technology in enterprises.

Keywords: end-user development, human-computer interaction, business process management, workflow modeling, computer-supported cooperative work, knowledge management.

Zusammenfassung

Geschäftsprozessmanagementlösungen ermöglichen es Unternehmen ihre Geschäftsabläufe zu konsolidieren und zu optimieren, und dadurch einen Wettbewerbsvorteil in der sich schnell entwickelnden Marktumgebung zu erzielen. Sehr oft sind die einzigen Personen, die Geschäftsprozesse im Detail kennen, die Endanwender von Unternehmenssoftware, die diese Geschäftsprozesse im Rahmen ihrer täglichen Arbeitspraxis ausführen. Hierdurch entsteht die Notwendigkeit, Endanwender in die Komposition von Geschäftsprozessen während der Implementierung von Geschäftsprozessmanagementlösungen in Unternehmen einzubeziehen. Endanwender besitzen eine detaillierte Fachexpertise, aber nur begrenzte technische Fähigkeiten. Eine explizite Prozessmodellierung durch konventionelle Modellierungssprachen und -umgebungen ist den Endanwendern selbst daher nicht unmittelbar möglich. Anwenderzentrische Ansätze zur Geschäftsprozesskomposition werden benötigt, die Endanwender in der Lage versetzen, Geschäftsprozesse entsprechend der eigenen Fachexpertise und Problemlösungsstrategien zu modellieren und anzupassen. Darüberhinaus müssen derartige Ansätze das Prozessverständnis von anwendungsorientierten Endanwendern, sowie technisch orientierten Prozessdesignern und Entwicklern im Rahmen von Workflow-Projekten verknüpfen. Dadurch können Geschäftsprozessmodelle erzeugt werden, die sich eng an den Bedürfnissen der Endanwender orientieren, als auch die technische Implementierung von Geschäftsprozessmanagementlösungen in Unternehmen erleichtern.

Diese Dissertation adressiert die endanwendergetriebene Komposition von: (i) schwach strukturierten Prozessmodellen zur Unterstützung unterspezifizierter, personenzentrischer Geschäftsprozesse und (ii) strukturierten Prozessmodellen zur Automatisierung von gleichartig wiederkehrenden Prozessen in Workflow-Systemen. Beide Prozesstypen lassen sich in einem kollaborativen Aufgabenmanagementsystem durch einen Programming by Example Ansatz komponieren. Das Aufgabenmanagement wird hierbei als Startpunkt für die endanwendergetriebene Prozesskomposition genutzt, um die individuelle Sicht der Anwender und die globale Sicht des Unternehmens auf Geschäftsprozesse in Einklang zu bringen. Programming by Example ist eine End-User Development Technik, welche die Erfassung und wiederholte Ausführung von Anwenderaktivitäten in einem Softwaresystem ermöglicht. Die Übertragung dieser Technik auf den Unternehmenskontext zur Komposition von Geschäftsprozessmodellen ist neuartig und erfordert eine spezifische Unterstützung der Endanwenderperspektive, sowie eine angepasste Ausgestaltung formaler Systeme.

Die vier hauptsächlichen wissenschaftlichen Beiträge der Dissertation sind: (i) ein Aufgabenmanagementmodell für personenzentrische Geschäftsprozesse; (ii) eine Methode zur Komposition von schwach strukturierten Prozessmodellen durch kollaboratives Aufgabenmanagement; (iii) eine Methode zur Transformation von schwach strukturierten Prozessmodellen zu strukturierten Workflows und die Erweiterung der letzteren auf der Basis von Abweichungen durch benutzerdefinierten Aufgaben, die während der Workflow-Laufzeit entstehen können; (iv) ein holistisches Konzept zur endbenutzergetriebenen Geschäftsprozesskomposition durch Programming by Exampe, das die Beiträte (i) bis (iii) in eine ganzheitliche Methode und Architektur für die Komposition von schwach strukturierten und strukturierten Prozessmodellen zusammenführt. Die ausgearbeiteten Konzepte bilden einen signifikanten Beitrag zu bekannten Prozessmodellierungsansätzen aus verschiedenen Forschungsgebieten, wie Mensch-Computer-Interaktion, Geschäftsprozessmanagement, Workflow-Management und computerunterstützte Gruppenarbeit.

Die ausgearbeiteten Konzepte basieren auf Erkenntnissen durchgeführter empirischer Studien. Diese umfassen einerseits einen Onlinefragebogen, der an Unternehmen aus verschiedenen Industriezweigen verteilt wurde. Andererseits umfassen die empirischen Studien eine Serie von Feldstudien in drei kleinen und mittelständischen Unternehmen in Deutschland. Die empirischen

Vorstudien liefern wichtige Erkenntnisse im Hinblick auf bereits bestehende Praktiken des End-User Developments im Bereich des Aufgabenmanagements und identifizieren Problembereiche durch deren Adressierung Endanwender sinnvoll in die Prozesskomposition einbezogen werden können. Die Studienergebnisse erlauben die Ableitung grundsätzlicher Anforderungen, die durch das entwickelte Aufgabenmanagementmodell adressiert werden und spielen eine entscheidende Rolle für die Ausgestaltung der Architektur im Rahmen des holistischen Konzepts.

Die entwickelten Konzepte berücksichtigen sowohl die existierenden Arbeitspraktiken der Endanwender, als auch die von ihnen verwendeten Softwareapplikationen, die sich auf das Management ihrer täglichen Arbeit beziehen. Solche Applikationen sind zum Beispiel Anwendungen zur Verwaltung von E-Mails und persönlicher Aufgabenlisten, die oftmals als Teil von konventionellen Office-Anwendungen zur Verfügung gestellt werden. Das Aufgabenmanagementmodell ermöglicht eine Aggregation von Daten aus diesen Applikationen, die im Rahmen der Komposition von schwach strukturierten Geschäftsprozessmodellen verwendet werden können. Diese Modelle können zur Ausführung von unterspezifizierten, personenzentrischen Prozessen bei Bedarf wiederholt angepasst und wieder verwendet werden. Um gleichartig wiederkehrende Abläufe durch Workflow-Systeme automatisieren zu können, wurde eine Methode zur Erzeugung von strukturierten Workflows auf Basis der erfassten, schwach strukturierten Prozessmodelle entwickelt. Erzeugte Workflows können von Prozessdesignern und Entwicklern in einem Kontext erweitert werden, in dem sowohl die benutzerdefinierten als auch die formellen Prozessmodelle gleichzeitig verfügbar sind. Die Zuordnung von schwach strukturierten zu formellen Prozessmodellen erleichtert die Wiederverwendung von benutzerdefinierten Prozessdaten. Der Datenaustausch und die Zusammenarbeit zwischen ad-hoc Prozessen und strukturierten Prozessen werden durch das holistische Konzept und die darunterliegende Architektur ermöglicht, die im Rahmen der vorliegenden Arbeit entwickelt wird.

Die dargestellten Konzepte wurden in einem Prototyp namens Collaborative Task Manager implementiert und evaluiert. Die Evaluationsergebnisse bestätigen, dass der entwickelte Prototyp durch die ausgearbeiteten Konzepte die endbenutzergetriebene Geschäftsprozesskomposition in der betrieblichen Praxis effektiv zu unterstützen vermag. Die im Rahmen dieser Arbeit entwickelten Konzepte lassen sich daher zum Design neuartiger, benutzeranpassbarer Geschäftsprozessmanagementsysteme verwenden, die sowohl eine breitere Nutzung, als auch eine leichtere Anpassung von Geschäftsprozessmanagementtechnologien in Unternehmen ermöglichen.

Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER 1: Introduction

Information Technology (IT) has conquered the workspace of business users over the last decades and provides vital support for personal and group activities in enterprises. The palette of indispensable software applications ranges from common office tools such as email [DB01], spreadsheets [NM90] and word processors [Esk05], to complex enterprise resource planning systems [Dav98] and business process management systems [vdAHW03, Gad08] for supporting business operations in cross-functional business areas.

While software systems keep enterprises running, they determine also the extent to which enterprises are able to adapt to changes in the business context, arising e.g. from external market conditions, legal requirements or partner relationships [WR95]. Challenges and pitfalls for enterprise systems with this respect lay particularly in the need to reconcile the technological imperatives of enterprise systems with the actual business needs of the enterprises [Dav98]. Business process management systems face a particular challenge in the need to reconcile automation support with flexibility [AS94, RRMvdA05]. Flexibility is an issue also for standard office applications such as email [BDH+05] and word processors [PJAA96], which need to respond to personal preferences of end users in order to increase the individual performance.

Yet, from formal systems' design perspective it is impossible to anticipate all requirements that will result from the usage of a software system by various end users, with different business domain expertise and technical skills, in different business contexts. In order to make enterprises more flexible and to increase the economic expectations from IT investments, software systems need to be adaptable in the *"context of use"* by their actual end users [WJ04]. This requires software systems that are not only *"easy-to-use"* but also *"easy-to-develop"*, i.e. systems which incorporate enhanced End-User Development (EUD) capabilities [LPKW06]. EUD is defined as *"a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify, or extend a software artefact"* [LPKW06]. EUD generally aims to provide a holistic view on the adaptation of software systems by end users and on user-centric system design, by exploiting synergies between related concepts and research fields.

The need for increased system flexibility and adaptability is especially relevant for the domain of Business Process Management (BPM) where agility is perceived as a mandatory requirement for businesses [For06]. BPM is a holistic management approach that promotes effectiveness and efficiency in value-adding business processes while enabling process alignment with company strategies, shaping processes from organizational point of view and adopting appropriate communication and workflow systems to support process management and automation [Gad08]. The basic premise in this thesis is that enhanced EUD towards the composition and adaptation of business process models by end users can enable adaptive BPM and make enterprises more flexible in the constantly changing business environment. Within the thesis a business process model is considered as a non-trivial software artifact, which is composed and adapted by end users in the sense of the EUD definition above. The adopted terminology is discussed more precisely in the next section.

1.1 Basic Terminology

The concept of a business process plays a central role in the dissertation. Various definitions of a business process are available in related literature. [DS90] defines a business process as *"as a set of logically-related tasks performed to achieve a defined business outcome"*. [vdAvH02] further considers that every piece of work relates to handling of a specific *"case"*, like e.g. processing a

tax declaration, an insurance claim, producing a product etc. (cf. also [vdABV+99]). Thereby each case is handled through the execution of a given process. According to [vdAvH02] *"a process consists of a number of tasks which need to be carried out and a set of conditions which determine the order of the tasks"*. Further, [vdAvH02] defines a task as *"a logical unit of work which is carried out as a single whole by one resource"*, where *"a resource is the generic name for a person, machine or group of persons or machines which can perform specific tasks"*.

While the above definitions consider "tasks" as building blocks of a business process, according to [Wes07] *"a business process consists of a set of activities that are performed in coordination in an organizational and technical environment. These activities jointly realize a business goal"*. Thus related literature on business processes uses the terms "task" and "activity" with overlapping meaning, pointing at fine-granular building blocks of business processes. Further, [Wes07] considers that a business process model consists of a set of activity models and execution constraints between them. Thereby *"a business process instance represents a concrete case in the operational business of a company, consisting of activity instances. Each business process model acts as a blueprint for a set of business process instances, and each activity model acts as a blueprint for a set of activity instances"* [Wes07]. Considering the business process terminology discussed above the thesis adopts the following definitions:

Definition 1.1: (Business Process) A business process consists of a set of tasks that are performed in coordination in an organizational and technical environment. These tasks jointly realize a business goal.

Definition 1.2: (Task) A task is a self-contained, logical unit of work, which is carried out as a single whole by a given person, machine, group of persons or machines by using appropriate resources.

Definition 1.3: (Resource) A resource is used for the execution of a given task or generated as output from a task. The resource can be a person, machine or group of persons or machines which can perform a given task, but also a document or a tangible object that is required for performing a given task or that is produced or modified during the task execution.

Definition 1.4: (Business Process Model) A business process model consists of a set of task models and acts as a blueprint for a set of business process instances.

Definition 1.5: (Task Model) A task model describes a task and acts as a blueprint for a set of task instances.

Definition 1.6: (Business Process Instance) A business process instance represents a concrete case in the business of a company, consisting of task instances.

Definition 1.7: (Task Instance) A task instance represents a task in a concrete case in the business of a company and can be created from a task model.

The term "task" is adopted due to the close relationship between end-user driven business process composition and task management in the thesis. The definitions provided above do not differentiate between ad-hoc and structured process models and instances, or ad-hoc and structured task models and instances. In the thesis such differentiation is made where needed to avoid ambiguities.

1.2 Challenges for BPM Systems

Enterprises are constantly trying to optimize their business processes in order to gain competitive advantage in the fast evolving global market. For this purpose BPM strategies are developed and applied. A crucial aspect thereby is the adoption of appropriate BPM technology.

The shift from data-orientation, which dominated the software industry in the 1970s and 1980s, to process orientation in the 1990s, led to the development of Workflow Management Systems (WfMS). Related literature defines a workflow as *"the automation of a business process, in whole or in part, during which documents, information, or tasks are passed from one participant to another for action, according to a set of procedural rules"* [Wes07]. The purpose of a WfMS is thereby to manage the sequence of work activities and the invocation of appropriate human and IT resources associated with the various activity steps hence providing procedural automation of a business process [Hol95, vdAvH02, Wes07].

In related literature [vdAvH02] the term "workflow" is used as a synonym for "business process". Similarly, in the thesis the term "workflow" is used as a synonym for an "operational business process" which can be automated through a WfMS.

With the increasing power of information technology over the last years, new requirements for business process support emerged. This expanded the technological foundation provided by WfMS towards BPM systems. While WfMS focus predominantly on three phases of process automation: process design, system configuration, and process enactment, BPM systems enable additionally enhanced: process diagnosis, simulation, verification, and validation [vdAHW03]. The latter study defines a BPM system as *"a generic software system that is driven by explicit process designs to enact and manage operational business processes"* [vdAHW03]. Hence, BPM systems are generally considered as an extension that goes beyond WfMS and provide more comprehensive support for the management of operational processes [vdAHW03].

While related literature on BPM systems [vdAHW03, Wes07] focuses on operational processes and leaves out processes on tactical level and such processes that cannot be explicated, the thesis considers that software support for BPM needs to address also the latter process types. This presumption is found in related literature discussing the challenges for next generation BPM systems [RRMvdA05]. Hence, the thesis considers BPM systems as systems that support BPM by addressing different process types: ad-hoc, semi-structured, and structured. The thesis adopts the following definition of a BPM system (cf. also [Wes07]):

Definition 1.8: (Business Process Management System) A business process management system is a generic software system that is driven by explicit process representations to enact and to manage business processes.

In contrast to the definition of a BPM system given in [vdAHW03], the adopted definition leaves out the term *"operational"* by expanding the scope to generic business process support, including ad-hoc and semi-structured processes. This broad scope is discussed in related work on Process-Aware Information Systems (PAISs). A PAIS is defined as *"a software system that manages and executes operational processes involving people, applications and/or information sources on the basis of process models"* [DvdAtH05]. The latter definition closely relates to the definition of BPM systems given in [vdAHW03] where the focus is set on operational processes. Nevertheless, PAISs consolidate research from different fields and apply a more generic view incorporating different types of business processes (cf. [DvdAtH05]): (i) person-to-application processes, addressed by WfMS, (ii) person-to-person processes, which are subject to Computer Supported Cooperative Work (CSCW) research, and (iii) enterprise application integration and business-to-business integration processes. PAISs hence provide a broad view on software support for BPM and exemplify the wide research scope related to business process composition.

While known research on WfMS, BPM systems and PAISs focuses predominantly on processes that can be explicated and relate to concrete process models, few has been done to investigate how the utilization and adaptability of such systems can be facilitated by involving end users in business process composition and how the appropriation of process models can be rendered to the end users (cf. [WJ04]). Some intrinsic challenges with this respect are evident from related literature, and addressed in this thesis.

1.2.1 Challenge 1: Supporting Underspecified, Human-Centric Business Processes

While conventional workflow solutions are well suited for static, predefined processes, they are unable to support knowledge-intensive, human-centric business processes, which are executed in distributed teams in a rather informal, ad-hoc manner [AS94, vdABV+99, Ber00, SAMS01, Jor04]. A detailed investigation of different aspects of enterprise efficiency related to knowledge work is presented in [Wii04]. The latter study clearly accentuates that the enterprise performance is a result from the individual actions of all involved employees. However, when discussing the challenges for next generation BPM and task management, [RRMvdA05] raises the issue that *"knowledge workers often concentrate on their tasks, forgetting the organizational needs of streamlining processes"*. The need arises to reconcile the personal task management perspective, and the enterprise BPM perspective into a common understanding of process. This novel view on business processes emerges in analyst reports as the *"Process of Me"* and *"introduces a new way of thinking of process — from the individual out, rather than from the traditional "enterprise in" model"* [Gar06]. This view is recognized as one of the major challenges for the next generation BPM systems as it states the fundamental need to provide end users with adequate techniques to proactively express process knowledge and to participate in business process management and design according to their actual expertise and problem solving strategies.

1.2.2 Challenge 2: Involving Business Users in Formal Process Modeling

Rigidly recurring processes are suitable for automation through conventional WfMS. However, workflow projects often suffer from inconsistencies, resulting e.g. from *"projecting the sequence of an interview onto real work situations or by assuming logical dependencies which do not correspond with reality"* [Her00]. The need arises to bridge the business and technology perspectives on enterprise processes by increasing the *"business collaboration in process modeling"* [For06] and enabling business users, process designers and developers, to work together on the elaboration of process models, i.e. in a shared process composition context. As a result, standardized graphical notations such as the Business Process Modeling Notation [OMG06] have emerged. Visual process modeling is offered in enhanced solutions by leading software vendors like e.g. IBM, TIBCO, Appian and others. However, achieving process support that is better turned to users' needs and organizational changes by *"letting end-users do the tailoring"* demands *"both domain expertise and advanced skills in computer use"* [MM00]. Upfront process modeling hence remains inaccessible for business users, who have good domain knowledge but limited technical skills. Such modeling can furthermore result in overhead for business users as it can be hardly considered as part of their daily activities. Studies on ad-hoc process support consider this limitation and suggest *"the existence of a separate organizational unit for process modeling"* [HMBR05], yet confirming the disruption between end users and business technology experts. The need for user-centric approaches arises, which can enable *"informed participation"* [FGY+04] of end users in business process composition without confronting them with upfront process modeling or deviating their focus from their daily business activities, and which can in the same time enable process tailoring as collaboration [MM00] between end users, process designers and developers.

1.2.3 Challenge 3: Enabling Adaptive BPM through User-Tailorable Process Definitions

While BPM technology can increase enterprise performance, a tradeoff is always considered between the related IT investments and the perceived benefits from BPM software [Ver04]. This tradeoff is especially critical for small and medium enterprises which generally have limited human resources and time for education. Such enterprises cannot adopt BPM technology if it comes with costly and time consuming external consulting. BPM software will hence add value, if it provides process definitions that can be tailored by the end users within the *"context of use"* according to the requirements of the evolving business processes rather than by *"the software vendor, external consultants, or in-house development team"* who are *"not involved in the business processes and do not share the respective work practices"* [WJ04]. Thus BPM systems need to incorporate EUD capabilities that allow enterprises to respond to the dynamically changing internal and external conditions by adapting their processes and organizational structure on-demand. For enabling adaptive BPM the thesis suggests that users should not only be involved in business process composition and formal modeling but also enabled to refine process models during process execution in evolving business contexts.

1.3 Objectives and Contributions

The thesis is motivated through the discussed challenges for BPM systems. The generic aim is to provide a framework that resolves these challenges by achieving the following objectives:

- *Enabling end-user driven composition of weakly-structured business process models for supporting work coordination and guidance in ad-hoc, human-centric business processes;*
- *Involving end users in formal process modeling without confronting them with upfront process modeling notations and environments;*
- *Enabling a shared context between user-defined and formal process models, where process designers and developers can refine formal models by referring to real-life process data;*
- *Enabling on-demand extension of workflow models based on user-defined deviations from structured workflow instances with unplanned, ad-hoc tasks;*

The thesis focuses on end-user driven composition of business process models, i.e. on the process modeling aspect. To achieve the above objectives the following concepts have been developed, which represent the original scientific contributions of the thesis.

1.3.1 Task Management Model

A task management model is defined as a formal specification of a set of concepts and relations that allow aggregation and handling of data for end-user driven business process composition based on personal task management. The notion of task as introduced in Definition 1.2 is used (cf. also [vdABV+99, vdAvH02]). Business processes are considered as composed of a predefined or ad-hoc sequence of tasks with associated resources and involved human actors. A particular focus is set on the aggregation of data from existing end-user software environment to increase the unobtrusiveness for process tailoring by end users. End-user driven process composition is thereby supported through light-weight, personal task management, both in the personal as well as in organizational settings. The task management model described in the thesis is the full, expanded version of previous work [SSS07]. It consists of two major, interrelated parts: (i) *runtime task management model*; (ii) *task pattern model*.

The *runtime task management model* describes the task management model at instance level, i.e. it describes concepts and interrelations that support composition of emergent, weakly-

structured process models during end users' task management activities in running ad-hoc processes. This model defines the attributes of task instances for capturing contextual task and process information. The model further defines entities and relationships for capturing the conversational flow for task delegation, which enables reasoning about the collaborative handling of ad-hoc tasks. Further, the model includes entities for capturing the event flow on ad-hoc tasks, which is evaluated during ad-hoc to formal process model transformation to determine the task sequence for the formal workflow model. A mechanism for interrelation of ad-hoc and structured processes is also contained. This mechanism enables the interoperability between formal and ad-hoc processes and the tailoring of formal workflows by end users through deviations with ad-hoc tasks at runtime. For aggregating process data the *runtime task management model* relies on input from personal task lists with to-do items and email, which are delivered with standard office applications. The utilization of these software environments is motivated through the preliminary empirical studies, presented later on in the dissertation.

The *task pattern model* describes the task management model at schema level, i.e. it defines task patterns [SSS07, RRMvdA05, GOR+07] as reusable task structures that serve as models (schemes) for producing ad-hoc task instances in concrete ad-hoc processes. Task patterns can be extracted from executed ad-hoc processes to capture process knowledge in reusable manner or they can be created from scratch during design time as explicit best-practice definitions. In case of task pattern extraction from ad-hoc process instances, the task pattern model supports exclusion of some of the runtime data of ad-hoc task instances, which is relevant only for a concrete execution case, but enables references to this data for later analysis. Task patterns can be adapted and reused in evolving ad-hoc processes. When a task pattern is applied, relevant attributes from the pattern are applied to the resulting task instance and enable guidance according to the explicit best-practice that is defined in the reused pattern.

Both models – the runtime task management model and the task pattern model, share some common attributes. Both models enable hierarchical task decomposition for a process description and management at different detail levels. Both models further provide binding of documents and of transactional applications into tasks through artifacts. Basic entities for human actors' representation are also contained in both models.

1.3.2 A Method for Composition of Weakly-Structured Process Models

The actual composition of end-to-end business process models through the underlying task management model is described through a method for composition of weakly-structured process models [SSS07, SSFM08a]. The method uses the task flow, document flow and human actor information, provided through the runtime task management model, and defines the binding of personal task hierarchies into end-to-end process models during the end users' task management activities. This binding is accomplished at process instance level by integrating individual task hierarchies of multiple process participants based on task delegation over email. The method defines also the collaborative handling of tasks by considering limitations for ad-hoc work coordination from the CSCW domain. The method discusses also basic adaptations of emerging ad-hoc processes in different scopes, affecting individual as well as collaborative tasks. A transition from a captured ad-hoc process instance to task patterns upon task pattern extraction is also defined by the method. On the other hand, the method defines the transformation of task patterns to task instances, when a task pattern is reused in an ad-hoc process. Tracing of evolutionary relationships between task patterns and task instances, and between different task pattern's variations resulting from task pattern reuse are also discussed in this method.

1.3.3 A Method for Composition of Structured Process Models

A method for composition of structured process models through transformation of weakly-structured, user-defined process models is provided. This method enables automation of rigidly

recurring processes through workflow engines [SSFM08c]. The method defines the transformation of the captured control and document flow as well as human actor information in terms of task assignments from ad-hoc to structured process models. This method uses the captured change history of ad-hoc tasks as defined in the task management model, to evaluate the ad-hoc task sequence and to generate appropriate control flow for the formal workflow models. The transformation is based: (i) on the hierarchical order of ad-hoc tasks, (ii) on the task delegation flow, and (iii) on the temporal relationships between changes in ad-hoc tasks that alter specific task attributes. A transformation scheme for associated artifacts (documents) from ad-hoc processes into formal workflows is also provided. The transformation of task assignments is discussed with respect to established workflow modeling notations. Thereby a generalization of ad-hoc task associations to human actors is proposed, which enable role-based task assignments in the derived structured workflows.

The method further describes the extension of structured workflows based on ad-hoc task deviations at runtime [SSFM08d]. The method proposes basic workflow task states and defines the interrelation between deviating ad-hoc tasks and running workflow task instances. Based on that, the method provides rules for the embedding of deviating ad-hoc task hierarchies into an originally derived workflow model for its redesign according to the evolved business context.

1.3.4 Holistic Concept for End-User Driven Business Process Composition

The presented holistic concept composes contributions (i) through (iii) into a seamless overarching method and architecture for the composition of weakly-structured and structured process models. Two major aspects are considered: ad-hoc process support through process-enhanced task management [SSFM08b], as well as ad-hoc to structured process model transformation under increased data reuse and interoperability between ad-hoc processes and structured workflows [SS08a]. The holistic concept describes the different aspects of user-centric process support that are used to gradually involve end users in business process composition. End users are enabled to extend their skills with conventional applications for task management and collaboration such as to-do lists and email towards the composition of weakly-structured and structured process models. The motivation for this skill acquisition is added value on personal task management. The added value is provided in various directions such as transparency and reuse of previous knowledge in evolving collaborative processes. The gradual involvement of end users in business process composition is considered in the presented system architecture, which supports different aspects of personal and organizational task management through different system components. For ensuring unobtrusive support for process tailoring, the architecture enables integration of the process composition environment into the existing end users' application environment. The architecture defines further the system components that are used to aggregate data according to the introduced task management model and to generate process models according to the process composition methods. The architecture consists of three tiers. It exposes a light-weight, process-enhanced task management client for composition of ad-hoc business processes, editing of task patterns, and generation and adaptation of structured workflows. A middleware is provided which encompasses all services, responsible for data aggregation, distribution and retrieval. The provided persistence tier comprises a set of repositories for storing process data in terms of task (control) flow, document flow and user information. The holistic concept defines how the services from the middleware interoperate and share repository data and what functionalities are exposed to the users in order to hide complexity and increase their tailoring abilities.

1.3.5 Practical Contribution – Collaborative Task Manager (CTM)

While the task management model, process composition methods and system architecture are conceptual contributions of the presented thesis, it has generated also a significant practical

contribution – the Collaborative Task Manager (CTM) prototype [SSFM08a, SSFM08c]. CTM is an advanced task management system, which enables: (i) light-weight composition of weakly-structured process models for ad-hoc process support; (ii) formalization of weakly-structured process models to structured workflow models for automation of rigidly recurring processes. CTM is a technical realization of the presented concepts, incorporating broad technological foundation. The system tracks user activities on personal task management in local to-do lists and replicates task data to a central server instance. Email exchange for task delegation is tracked to interconnect the individual task hierarchies of different process participants to end-to-end enterprise processes on the server. Through this CTM enables transparency in evolving collaborative processes beyond the capabilities of standard to-do list and email applications. CTM further realizes the task pattern concept and enables creation, extraction, adaptation and reuse of ad-hoc task and business process models without confronting end users with formal task or process modeling notations. Formalization of weakly-structured process for the generation of structured workflows is enabled directly in the to-do list and email environment. Through this a shared context is provided between user-defined task hierarchies and derived structured workflow models. This context enables local developers and business technology experts to edit the formal workflow models by referring to the real-life data of the original ad-hoc process instances.

1.4 Research Methodology

The research method of this thesis is based on the incremental development cycle [Gra89, LB03] where requirements determine the concepts and the design choices for a software system, and its usage generates new requirements. Thereby validation is integrated through the implementation, which shows that the elaborated concepts and the realized design are feasible and that the system fulfils the initial objectives.

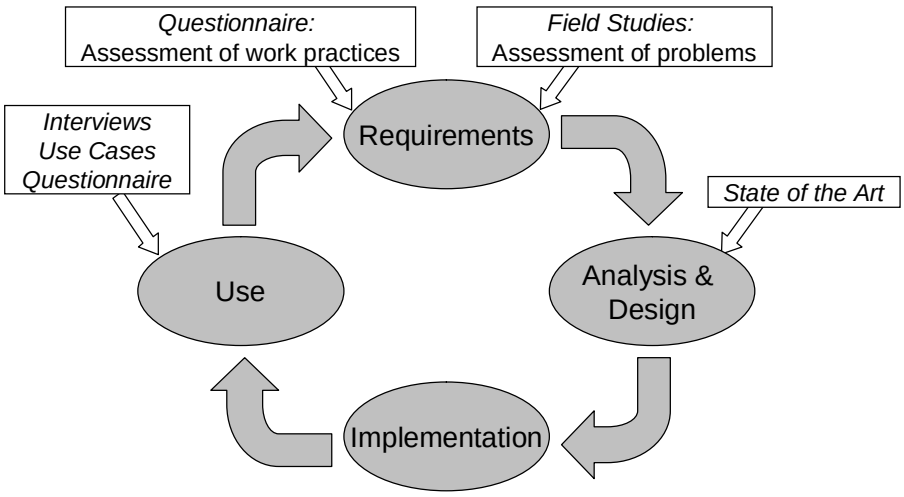


Figure 1.1: Research approach overview

Concretely, requirements presented in this thesis result from user studies, which consist of two major parts. The first part comprises an online questionnaire that assesses existing end user work practices related to task management and associated tailoring activities. Task management has been specifically selected because of the need to reconcile the personal task management and the organizational BPM perspective on enterprise processes [RRMvdA05]. The second part comprises field studies at three small and medium enterprises and provides an assessment of

existing problem areas in managing and coordinating informal business processes. Both studies have been conducted independently. The studies result in a set of requirements for enabling end-user driven business process composition (see Figure 1.1). The applicability of existing approaches with respect to the findings from the empirical work is evaluated in the analysis and design phase to elucidate the deficiencies in supporting end-user driven process composition and to justify the need for the developed concepts. A conceptual framework for end-user driven business process composition is elaborated. This framework underpins the user-centric design of a system for process composition by end users according to the identified problem areas and to the state of the art analysis. A technical implementation of the conceptual framework is realized in a prototype system. The evaluation of the developed concepts is performed through practical application of the system in a real-life, enterprise context. A preliminary, qualitative evaluation is performed, which comprises two weeks of system usage followed by a set of interviews and contextual enquiries [BH98]. This evaluation delivers first user feedback and reveals additional requirements towards end-user driven process composition. Some of the requirements are implemented and delivered with a second prototype version. A long term evaluation is further performed comprising several weeks of prototype usage. This evaluation phase is also qualitative and concludes with a set of contextual enquiries [BH98] and interviews. The evaluation results are described in a set of use cases, which exemplify how the research objectives are accomplished through the introduced software system, i.e. through the realization of the provided conceptual framework. The overall evaluation concludes with a questionnaire-based assessment of the potential acceptance of end-user driven business process composition. The questionnaire-based evaluation is designed according to the Technology Acceptance Model (TAM) [Dav85, Dav89] and assesses the various concepts for end-user driven business process composition by focusing on different aspects of the provided implementation.

1.5 Research Scope

In order to avoid ambiguities, this section introduces some basic terms that are used throughout the thesis and clarifies the research scope. The research presented in this dissertation was conducted as part of the project **E**nd **U**ser **D**evelopment **i**n **S**mall and **M**edium **E**nterprise **S**oftware Systems (EUDISMES) [EUD06]. The purpose of the project is to develop innovative EUD techniques for small and medium enterprises, which are gaining importance for the German software market. These techniques aim at enabling end users to manage and adapt the software infrastructure in organizational and process-related perspectives.

1.5.1 Addressed User Types

While an *end user* is generally the expected user, i.e. the target user that will operate a software system, the plethora of software systems, application domains and anticipated system users broadens the scope of the term *end user* immensely and refers to people with highly varying technical skills and domain expertise. For example, an end user of Microsoft Visual Studio or Eclipse is a software developer, whereas for a Microsoft Office application, like e.g. Outlook, the end user can be practically anyone who needs email, calendar or to-do lists. There is an ongoing debate in End-User Development (EUD) literature about the classification of end users. Some studies differentiate between two generic types of users - *"beginning users"*, who start to learn how to use application software, *and "professionals in diverse areas outside of computer science, such as engineering, medicine, graphic design, business, and more, who are not professional programmers"* [LPKW06]. Obviously, this definition excludes a software developer from the end user type. A common notion of different user types with respect to user-tailorable systems is established through [MCLM90, Mac90, NM90, GN92], which generally introduce three user

types: *end users*, *local developers*, and *professional programmers*. Thereby the *end user* type appears as a *"worker"* in [MCLM90] and identifies users of a software system who just want to get their job done and who are not interested in the system itself and do not have expectations of being able to tailor the system. This type of users is referred to as *"non-programmers"* in other related work [SDW08]. Different terms are used also for the *local developer* type, which in [MCLM90] appears as *"tinkerer"*, in [Mac90] as *"translator"* and in further related literature such users are referred to as *"super users"* [MM00]. This type basically refers to a *"worker who enjoys exploring the computer system, but may not fully understand it"* [MCLM90], i.e. to an end user, who may engage in system tailoring by extending their software skills. The *professional programmer* user type is self-explanatory. Programmers have the most tailoring power as they have the expertise to change entire system components or create new software from scratch.

This thesis aims to provide concepts for composition of process models by business users. It hence focuses on the *end user* and *local developer* types. End users are considered as the actual participants in enterprise processes. They can be for example employees from sales, purchase, accounting and management departments that have no or very limited IT skills. A *local developer* on the other hand can be e.g. an employee from the IT department, who deals with software systems to an advanced level but does not have process modeling or programming skills. An important point is that the considered *end users* and *local developers* are generally involved in knowledge-intensive activities, which may contain tactical tasks and require ad-hoc cooperation. Hence, the business users considered within this thesis fall into the domain of knowledge-workers [SAMS01, Wii04, RRMvdA05].

1.5.2 Addressed Business Process Types

Business processes may highly vary depending on the business domain and business goal of the processes [Wes07, Gad08]. The thesis focuses only on company-internal processes, which may require cooperation of multiple users and departments, but do not cross the enterprise boundary and do not require integration of external stakeholders. Hence, no business-to-business integration processes are considered. Processes may further require automated, transactional tasks that are performed by a system agent, e.g. embedded in a WfMS. The thesis focuses on informal, human-centric business processes that are not currently supported through a groupware or workflow application. These processes encompass sequences of interrelated manual tasks of multiple users, who need to cooperate and to coordinate their activities by using conventional task management an email applications provided with common office tools. Thus, the thesis addresses processes for which BPM needs to be enabled from scratch, starting from process emergence and reaching to process design, deployment and redesign [Ver04].

1.6 Structure of the Dissertation

The thesis consists of nine chapters. The structure is aligned with the research methodology presented in Figure 1.1. The different chapters are briefly summarized in the following.

Chapter 1 has outlined the problem areas, central objectives and contributions of the thesis. These underpin the discussion on end-user driven business process composition throughout the dissertation.

Chapter 2 presents the empirical foundations for the dissertation. It describes the purpose, design and method for the empirical studies that have been conducted to elucidate the problem domain and to form the basic requirements for end-user driven process composition.

Chapter 3 provides a state of the art analysis on business process composition. It identifies fundamental EUD concepts and evaluates the suitability of different EUD approaches for process tailoring by end users. An analysis of process composition approaches from different research

domains such as workflow management, BPM, CSCW and knowledge management is further provided, which is underpinned through the fundamental EUD concepts and the findings from the empirical work.

Chapter 4 presents the task management model that enables aggregation of process definitions from conventional task management and email applications towards business process composition by end users. The task management model is the first original scientific contribution of this thesis. The structure of the model is motivated through several fundamental concepts from related literature and through the findings from the empirical work

Chapter 5 introduces a method for composition of weakly-structured process models. This method defines how data is aggregated from the underlying email and task management applications to assemble end-to-end, user-defined process models. The method further defines how extraction, adaptation, reuse and analysis of user-defined models are enabled in the context of ad-hoc business processes.

Chapter 6 introduces a method for transformation of weakly-structured process models to structured workflows towards automation of rigidly recurring processes through workflow engines. The method further defines how formalized process models can be extended based on user-defined hierarchies of ad-hoc tasks that result as deviations from workflow instances.

Chapter 7 presents the holistic concept which composes the introduced task management model and methods into a seamless overarching method and architecture for the composition of weakly-structured and structured process models. This concept is the third major scientific contribution of this thesis. This concept enables gradual involvement of end users in business process composition. Such involvement is supported through the corresponding underlying architecture. The design choices for the underlying architecture are motivated through the findings from the empirical work and through related literature. The introduced holistic concept ensures unobtrusiveness through tight integration in the actual end users' working environment and enables enhanced data reuse between user-defined and formal process models.

Chapter 8 describes the implementation of a software system that realizes the discussed task management model, process composition methods and holistic concept. The prototype system is called Collaborative Task Manager and represents the practical contribution of this thesis. The most important system components that relate to the developed concepts are discussed.

Chapter 9 provides an evaluation of the elaborated conceptual framework based on the application of the Collaborative Task Manager prototype in real-life, enterprise context. First, preliminary results from initial test usage are presented, as well as a set of resulting additional requirements. A set of case studies is further presented, describing different scenarios of system usage identified after a long-term system application. The case studies focus on different aspects of process composition with respect to the introduced challenges for BPM systems and the objectives for end-user driven business process composition. The evaluation concludes with a questionnaire-based assessment of the developed concepts which is based on the assessment of the related major system components according to an established technology acceptance model.

Chapter 10 provides a summary of the dissertation. A critical discussion of the implications of the presented work for business process management and end-user development is further provided. Finally, the chapter gives an outlook for future work that concludes the dissertation.

CHAPTER 2: Empirical Foundations

Although the need for involving end users in business process composition is a key issue for BPM vendors, industry analysts and researchers, it cannot be assumed that end users are motivated to participate in process tailoring per se. The need arises to achieve user-centric process support by bridging the user and the enterprise perspectives on business processes [RRMvdA05], i.e. by resolving concrete user problems related to personal task management, and additionally delivering added value to the enterprise as a whole through enhanced BPM. Hence, approaches for end-user driven process composition need to be based on a thorough understanding of how people work and organize their daily activities, and what can deliver value to them. To prepare the ground for the conceptual work on end-user driven business process composition, a series of empirical studies have been conducted, which tackle exactly these questions.

2.1 Assessment of Current Work Practices

The generic assumption behind the presented study is that users are generally interested in getting their job done and primarily gain effectiveness from efficiently managing their individual tasks. Therefore the first step towards enabling end-user driven business process composition is to understand the users' intent to customize software artifacts related to their personal task management. Task management support is discussed in related literature especially with respect to ad-hoc, knowledge-intensive processes where the application of formal and rigid workflow management systems for the management and optimization of individual and group activities is inappropriate [RRMvdA05, HMBR05, HRD+06]. The ad-hoc nature of knowledge work implies that users are often required to act problem-oriented, by adapting their work practice and software environment to handle a specific business case. For example, public servants may need to set complex rules for bid procurement and authorization, which are different depending on the nature and size of the purchase. This brings to the fore issues of effort and reuse. The required combination of flexibility and control often means that business users, who are not programmers, have to cross the boundary from adjusting software parameters to writing complex rules to change or enhance their application. End-User Development (EUD) literature considers customization and parameterization as initial EUD activities, whereas complex rule definition, e.g. through a scripting language, is even regarded as a programming activity [LPKW06, Bla06]. Crossing the boundary from parameterization to programming increases the cognitive effort required from end users. This makes their decisions about using programming techniques more sensitive to the "economic" tradeoff between costs (learning effort, perceived risks, etc.) and benefits (perceived increase in effectiveness, potential for recognition by peers, etc.). These tradeoff factors are referred to as *"EUD economics"* and relate to previous work on EUD benefits and risks [MSL06].

This section describes and analyzes findings from a detailed questionnaire conducted by the author of the thesis with together with members of the NEPOMUK project [NEP06] and the Manchester University of Technology [MSG+08]. The questionnaire has the purpose to explore existing EUD practices, to gauge end users' perceptions of EUD risks, benefits and proposed supporting actions and to identify factors which facilitate EUD practices in the domain of task management. This domain has been chosen because of its particular relevance for supporting collaborating business users in the context of knowledge-intensive, human-centric business processes [RRMvdA05, HMBR05, HRD+06]. Supporting such processes refers to the first major challenge for BPM systems addressed in this thesis (cf. Section 1.2.1).

The study addresses user groups from industrial companies in USA and Germany. The

reported findings are shaped by the main hypothesis that task management takes place in small- to mid-sized collaborative groups and that it represents a suitable candidate for EUD activities. The main aim of the survey is to find out what are the existing EUD attitudes and practices for task management, and to discover the suitability of task management for EUD. The survey results provide some insights about the feasibility of aligning individual and enterprise performance by enabling support for informal business processes through EUD-enhanced task management.

2.1.1 Background

The literature reports different technology-focused approaches for enabling user-centered task management. These range from simple work organization in personal to-do lists [BDG+04] to task-centric support in email environments [BDHS03] and business-process oriented task management allowing proactive information delivery and process know-how reuse [HRD+06]. While these studies target at concrete real-life problems, they do not consider generic users' intentions to engage with information technology at an increased level of complexity, i.e. towards tailoring software artifacts related to personal task management. Studies on technology adoption and use in social context [Suc87, OG94, Nar93] clearly exemplify that peoples' intentions to use information technology are strongly influenced by their individual interpretation of this technology and by the social environment in which this technology is used. An important role thereby plays the collaboration between people with different technological skills and business domain expertise. Technology is seen as a necessary but insufficient prerequisite for user empowerment in organizations [Nar93]. Therefore the only possible approach for estimating opportunities for end-user driven business process composition through user-tailored task management is through exploiting what are the actual users' work practices, benefit expectations and intentions related to EUD in the domain of task management, and what organizational prerequisites are there to support such EUD activities.

Organizational and personal factors that influence the decisions of individuals to use information technology are in the focus of technology adoption theories. A psychological perspective that can be used to assess individuals' intentions to engage with information technology is provided by the Theory of Reasoned Action [FA75]. This theory postulates that voluntary behavior of individuals can be predicted through their attitude towards the behavior and the individuals' perception of how other people would view them if they perform this behavior (subjective norm). This theory is extended through the Theory of Planned Behavior [Ajz85], which introduces the concept of self-efficacy in addition to attitudes and subjective norms. Self-efficacy relates to the perceived behavioral control, i.e. it expresses the conviction of individuals that they can successfully execute the anticipated behavior. The concept of self-efficacy originates from the Social Cognitive Theory [CHH99], where it is combined with the expectations of a valued outcome from a performed behavior. The Social Cognitive Theory introduces also behavioral motivation, resulting from individual's observations of other people performing (successfully) a certain behavior.

These psychological foundations are used to develop information systems theories for the acceptance and use of information technology. Such theory is the Technology Acceptance Model [Dav89], which is based on Theory of Reasoned Action [FA75] and replaces its attitude measures with two technology acceptance measures — perceived usefulness and perceived ease of use. Both theories have strong behavioral elements and postulate that when someone forms an intention to perform a certain action, they will execute this action, i.e. that intent is a key determinant of action. Further work extends the Technology Acceptance Model to account for social influences on individual behavior [MG99] towards bridging the technology-focused and psychological issues related to technology acceptance.

In the area of EUD, the Attention Investment Model [Bla02] is a theory related to technology acceptance, which is often applied to the cognitive design of EUD tools. The basic premise of this model is that the individual's decisions regarding the use of a tool feature or performing the

tailoring activity are driven by the balance between the cognitive costs of performing the tailoring activity and the benefits arising for the individual out of this activity. The impact of non-cognitive costs such as loss of personnel time on the balance of costs and benefits is recognized in [SLM03] where different types of EUD tools are classified according to the shape of their cost-benefit curves. Cost-benefit evaluations are further reported in [Sut05]. Hence, cost-benefit estimations are commonly used in the area of EUD for technology acceptance assessments.

While the psychological and information technology theories discussed above can help to estimate the factors influencing the individual intentions to perform EUD activities, EUD has also impact on organizational level [WJ04, FGY+04]. Therefore it is important to make a combined assessment of individual and organizational factors which can affect EUD uptake in enterprises. First steps in this direction are provided through a preliminary survey of attitudes to the costs and benefits of EUD in organizations [MSL06]. The list of benefits, risks and supporting actions introduced in the latter study underpin the presented questionnaire-based survey. The presented survey consolidates with related EUD research and puts forward the perceived balance of benefits and costs as one of the main factors impacting EUD uptake.

2.1.2 Questionnaire

The questionnaire was designed by fusing findings from technology adoption literature discussed in Section 2.1.1 with findings from a preliminary study on EUD benefits and risks conducted by the University of Manchester [MSL06]. The conceptual model, underlying the presented questionnaire is discussed in the following. A complete validation of this model is out of scope for the dissertation. For the purposes of the dissertation the questionnaire is used as an instrument to gather facts on existing EUD attitudes and practices for task management, and to discover the suitability of task management for EUD.

2.1.2.1 Underlying Model

The initial premise behind the presented questionnaire is that intent is a key determinant of action, for which strong evidence is provided in psychology as well as in information technology literature related to technology adoption [FA75, Ajz85, Dav89], together with two necessary pre-conditions: availability of resources and suitability of technology, the relevance of which is evident from previous work on EUD benefits and risks [MSL06]. The questionnaire hence considers EUD *Uptake* in organizations as a (yet undefined) function of the users' *Intent* to perform EUD, the availability of *Resources,* and the suitability of the *Technology*.

$$EUD\ Uptake = f(Intent, Resources, Technology) \quad (1)$$

Of these three determinants, the survey is focused on (a) establishing respondents' *Intent* to perform EUD; and (b) on availability of *Resources*. These two measures are used as predictors regarding the actual uptake of EUD. The *Technology* is explored only with respect to currently used tools and basic required features for task management support. *Technology* assessments thereby refer to the status quo of software support for task management. The development of sophisticated systems that are able to leverage EUD capabilities towards end-user driven business process composition is considered as subject to future work, which depends on the questionnaire results for the general predictors for EUD uptake – *Intent* and *Resources*.

Following the Social Cognitive Theory [CHH99], *Intent* is considered as depending on *Use context, Individual judgment* and *Background*:

$$Intent = f(Use\ context, Individual\ judgment, Background) \quad (2)$$

Thereby the *Use context* covers a number of organizational and community factors. The

Individual judgment covers the *Expectation of economic outcome* and *EUD self-efficacy*. The *Expectation of economic outcome* is driven by related work on benefits and risks of EUD [MSL06]. It includes personal-level factors (i.e. self-improvement vs. career sidetracking) and work factors (i.e. work effectiveness vs. impact from errors). *EUD self-efficacy* refers to the belief of respondents that they can perform EUD and the degree of their confidence in doing so. *Background* covers the presence of specific EUD and IT background plus psychological factors such as locus of control and need for cognition.

For the purposes of the survey *Resources* are measured as *Availability of spare time* and *Technical support* facilities.

Resources = f (Spare time, Technical support) (3)

The study is focused on the application domain of task management because of its particular relevance for supporting business users in the context of underspecified, human-centric business processes [RRMvdA05] and hence contains a significant number of questions about task management and general EUD experiences, in effect expanding the *Use context* and IT *Background* variables.

2.1.2.2 Setting and Layout

To assess the different aspects affecting EUD *Uptake*, the questionnaire was divided into six sections as described in the following.

Section 1 gathered information pertaining to two variables contributing to *Intent* (see (2)): *Use context* and *Background*, with questions exploring workplace and task management experiences. The questions were separated in several groups, focusing on the following major aspects: workplace - including company size and people in immediate proximity; amount of tasks and collaborative tasks; software (tool) support for task management; support for to-do lists; anticipated benefits from software support for task management with respect to given features. The questions regarding software support for task management from the last three question groups further provided basic information about the current *Technology* support for task management, without considering its appropriateness for EUD.

Section 2 elaborated on the IT-related *Resources* at work, a key prerequisite for *Uptake* alongside *Intent* (see (1)) and contained two groups of questions, following (3): Time for work improvement and IT support.

Section 3 contained some further IT *Background* questions, exploring user experiences with software on a more generic level such as usage of word processors, databases, spreadsheets and internet as well as programming and customizing applications.

Section 4 comprised questions focusing on EUD in a task management context. The section started with a simple scenario, pointing at several benefits of EUD for task management such as defining rules for information filtering and deadline monitoring, reusing past coordination templates and automated processing of emails related to a given task. The scenario was followed through several groups of questions focusing on the following aspects:

- *Use*, including actual *Uptake* or *Intent* to use EUD for task management;
- EUD *Self-efficacy* items, measuring the belief of the respondents that they can actually do EUD for task management, a key determinant of *Individual judgment* from (2);
- Attitudes towards *Benefits* from developing custom task management software and rules; and towards *Drawbacks* from EUD for task management; thus shaping *Expectation of economic outcome* which is part of the *Individual judgment* from (2);

Section 5 elaborated on actions in support of EUD and comprised two groups of questions focusing accordingly on the actions which may support the successful application of EUD and on the participant's intent to use EUD in their job after having considered benefits, costs and

supporting actions.

Section 6 concluded with further questions on the participants' *Background* (see (2)), such as age, gender, number of persons they supervise, number of IT related courses in formal education, job specialization and industry sector of employer.

2.1.2.3 Target Audience and Distribution

The target audience consisted of employees with different specialization and business background from companies in a variety of industry sectors. The majority of the participants were employees from industrial companies from the USA that are members of the Americas' SAP Users' Group (ASUG). The questionnaire was distributed randomly through the latter organization. The respective participants were contacted over email with a link to an online questionnaire, hosted on an external web site. Additionally, the questionnaire was distributed to twelve employees from the German sales department of SAP representing typical business users. They were addressed over email with a link to an online questionnaire on a company-internal web site. Both distributions were executed in parallel. In both cases there were no eligibility filters apart from membership of the target group of users. The participation was voluntary. A lottery with several attractive prizes was offered to the survey participants, to make sure that the participation will not be driven by their interest in IT and EUD. Thus the set of respondents was expected to be fairly heterogeneous in terms of IT skills, job specializations and education. Further, the online questionnaire forms contained input validation to avoid empty or incorrect answers.

133 persons participated in the survey. The underlying expectations about a heterogeneous target audience with respect to job specializations were largely confirmed (see Table 2.1) but still some weariness was implied through the slight IT-bias. To alleviate remaining concerns, the questionnaire followed the definition of EUD and included a question about the percentage of IT courses in formal education. For the purposes of the present analysis all 94 respondents which had less than 40% IT courses are considered as "IT naïve" and the remaining 39 as "IT educated". Using this distinction, a separate analysis on key issues was done, e.g. Table 2.4 and Table 2.9. Furthermore, tests for correlation between IT education and attitudes found no statistically significant relation between IT education and perceptions of benefits and drawbacks regarding EUD (the Spearman correlation test produced a correlation coefficient of 0.018 between IT education and the perception of EUD benefits, and -0.04 between IT education and perception of EUD drawbacks).

Table 2.1: Survey participants' job specialization

Job specialization	% of 133	Job specialization	% of 133
Logistics	20	Business Development	1.5
Operations / production /service	9.8	Quality Control	2.3
Marketing / sales	7.5	Procurement	1.5
Financials / controlling	11.3	Health	0.8
Human Resources	0.8	Business analyst	1.5
IT	43	Education	0.8

2.1.3 Results

The questionnaire responses were supplied as raw data in comma separated values format and merged using Microsoft Excel, and then loaded in SPSS (originally Statistical Package for the Social Sciences) for further statistical analysis. The analysis procedures are detailed below together with the analysis results.

First, an overview of the work context factors and task management experiences are given followed by general EUD experiences. This leads to analysis of the respondents' perception of intent, efficacy, benefits and drawbacks from EUD for task management. Finally, some aggregate findings are reported.

2.1.3.1 Use Context and Background Factors Related to Work and Task Management

The questionnaire started with an exploration of the tasks which end users have to accomplish at work. Here a small excerpt of the questions and results are presented, which hold the key messages for user-centric task management support.

The frequency distribution of results shown in Table 2.2 reveals that users mostly deal with collaborative tasks, which often involve more than 2 other persons. This implies an increased need for supporting collaborative tasks with multiple stakeholders. Still, the complexity indicators below indicate moderate complexity, with a typical task involving between 3 and 5 people and between 5 and 8 individual operations on that task by an involved stakeholder. This then can be a fruitful target for end-user development. A further interesting aspect is the increased usage of documents in tasks. This usage was explored by asking the participants to estimate, what percentage of their tasks use more than 4 documents. Less than 10 % tasks matching this criterion were reported by only 10, 5% of the participants.

Table 2.2: Tasks at work

From the tasks you do at work what % are collaborative, i.e. involve people other than yourself?	**N**	**%**
hardly any (<10%)	6	4.5
10-30 %	26	20
31-50 %	46	35
51-80 %	32	24
nearly all (>80%)	23	17
How many people do you reckon are involved in a typical collaborative task you do?		
1 other	5	3.8
2 others	35	26
3-5	64	48
6-10	20	15
>10	9	6.8
How complex is a typical task in which you are involved? As an indicator you can use the number of your operations, such as sending one e-mail, or making one phone call. Do not count operations done by other collaborators on the task.		
simple (2-4 op.)	19	14
average (5-8 op.)	65	49
involved (9-15)	30	23
very complex (>15)	19	14
On average, what is the percentage of tasks where you have to work with more than 4 documents?		
<10%	14	10.5
11-30%	43	32
31-50%	31	23
51-80%	27	20
>80%	18	13.5

In order to investigate the actual tool usage, a further set of questions focused on the software, which participants use to manage their tasks and to coordinate work with others – see Table 2.3. Specifying multiple tools was allowed. The results clearly point at email as the primary tool for managing daily work. When asked about the specific email software, 62 of 90 participants reported to use Outlook, 43 of 90 use Lotus Notes and 3 use other software such as Groupwise – again some users reported using more than one tool.

Table 2.3: Tool support

Do you use any software to manage your tasks and coordinate work with others?	**N**	**%**
No software, I do it all with pen, paper and telephone	11	8.3
I use email	126	95
I use instant messaging and presence notification software such as ICQ or Skype	56	42
I use calendar management software	103	77
I use an integrated personal information manager (e.g. MS Outlook, Palm device, etc.)	62	47
I use workflow software	44	33

The next step in the exploration of the users' work practices focused on the usage of more sophisticated features of the available tools. The results are shown in Table 2.4, where the percentage distributions refer to the 94 IT naïve respondents and to the all 133 respondents. The results reveal that the respondents, especially the IT-naïve ones (75%), are already undertaking advanced customizations e.g. by creating email-filtering rules and de-facto performing EUD activities.

Table 2.4: Advanced feature use

Does your e-mail tool allow you to define any kind of processing rules for filtering, forwarding or sorting messages?	% IT Naïve	% All
Don't know	4.3	5.3
No	2.1	1.5
I know it does but I don't know how to do it	19	20
Yes and I know how to do it	75	73
Does the software tool which you use for task management maintain a 'To Do' list of tasks?		
I don't know	12	9.8
No	3.2	3.8
Yes	35	33
yes and I have used it at least once	27	27
yes, I use it regularly	23	26

The responses show that users are aware of the possibility to organize and monitor their tasks with "to do" items and that they are making good use of such items. This fact points at the proactive attitude of end users towards organizing and managing work items.

The exploration of the users' task management experiences concluded with a brief assessment of the perceived benefits from several features of software support for task management (see Table 2.5). Results are generated from 133 valid answers on a 5-point ordinal scale: *Not useful for my work (1), Can be useful (2), Useful (3), Very useful (4), Vital for my work (5)*. In all tables, M stands for *Mean* and SD for *Standard Deviation*.

Table 2.5: Feature benefits

Feature	**Mode**	**M**	**SD**
Reminding me of task due dates	5	4.16	0.99
Showing tasks in order of their due dates	5	4.32	0.88
Showing tasks in priority order	5	4.32	0.83
Allows me to delegate tasks	4	3.82	0.97
Allows monitoring of the progress of tasks	4	3.92	0.96
Allows me to estimate task completion times	5	3.89	0.98

The results show that task management tool support can provide value to end users if it is able to support the above features by integrating different task views (task order filtering) and providing extended collaboration support, e.g. through enabling task delegation and overview in collaborative processes, plus capabilities for estimating task completion times, e.g. based on knowledge of previous task executions.

2.1.3.2 Resources

Information was gathered about the time available for improvement (see Table 2.6) and technical support (see Table 2.7), two constituents of the *Resources* variable. Two of the questions about time used ordinal scale so for these only % distributions are reported. Evidently the mode of time for self-improvement per day is between 15 and 30 minutes, whilst the mode for days in training is 2 to 5 days. The first question in the table tests if organizations would allow more time for EUD training. This is set in Likert scale [Lik32], and the result is 0.77 (Agree). The reported times and organizational support for EUD seem adequate for undertaking limited EUD activities focused to the needs of the respondents' daily work duties.

Table 2.6: Resources available

	M	**SD**
My organization would allow me to spend more time on this type of activities if I make a suitable case. A Likert scale ranging (-2,2)	0.77	0.77

What is the average time per working day available for you to improve your work organization or productivity? % of answers		Overall, how many days per year do you spend on your professional development? % of answers	
No time for coffee	7.5	Up to 2 days	19.5
<15 min	30.1	2 - 5 days	31.6
15-30 min	37.6	5-10 days	30.8
30 min -1 hour	15.0	11-15 days	8.3
>1 hour	9.8	>15 days	9.8

The reported agreements with two statements about IT support using Likert scale from (-2, 2) are shown in Table 2.7. The results indicate that IT support is believed adequate for the respondents' daily work and that they will be willing to help them with non-conventional queries.

Table 2.7: IT support

	Mean	**SD**
The IT support provided at my workplace is satisfactory for the needs of my daily work	0.77	0.87
The IT support people are unlikely to help me solve issues beyond the needs of my daily work	-0.22	1.14

2.1.3.3 Generic EUD-Style Activities

Section 3 of the questionnaire provided a generic introduction to EUD by exploring users' experience with software artifacts. Table 2.8 summarizes the relevant questions and the received answers. The questions were followed by a brief explanation, denoting that certain activities (shown in Table 2.8) can be classified as EUD and hence emphasizing on the transition from usage to customization and creation of software artifacts. The annotations (L) and (H) were not included in the questionnaire. These are used here to distinguish between activities which can be considered EUD activities under low- and high-threshold (or narrow and wide perspective) of

what constitutes programming. Because of the prevalent nature of entering simple formulae such as SUM in the spreadsheets (marked (N)), this activity was not included in the subsequent analysis.

Table 2.8: EUD activities

Tool	EUD activity	N	%
I have used the following features of a word-processor, e.g. MS Word:	Created macros using a macro recorder feature (L)	46	35
	Worked with the macro code (H)	30	23
I have used the following features of a database, such as MS Access:	Designed queries using visual designer (L)	67	50
	Designed queries using the SQL editor (H)	45	34
	Recorded macros (H)	37	28
	Worked with event code (H)	30	23
I have used the following features of a spreadsheet:	Entered formulae such as "SUM" (N)	130	98
	Entered conditional formulae (L)	113	85
	Developed data filtering rules(L)	116	87
	Recorded macros(H)	71	53
	Worked with the macro code (H)	45	34
I have used the following features of the WWW:	Used a package to create a database-driven website (H)	20	15
	Have developed sites in perl, Javascript, ASP, etc. (H)	19	14
I have performed the following programming operations: (H)	I have customized one-page programs	58	44
	I have developed one-page programs	50	18
	I have developed small software applications	47	35
	I have designed and developed software applications consisting of several interlinked routines	45	34
	I have designed complex applications	41	31

Table 2.9 compares the number of EUD activities reported by IT naïve users with those reported overall for the low- and high-threshold cases.

Table 2.9: Percentage of respondents engaged in EUD-type activities

Programming activities	L: Low threshold		H: High threshold	
	% IT naïve	% All	% IT naïve	% All
None	2.1	1.5	23	20
1-4	39	41	48	42
5-8	25	25	16	23
9-12	15	19	8.5	11
>12	11	14	4.3	4.5

As evident from these tables, EUD-type practices were widely reported, with only 23% of the "IT naïve" respondents not having done EUD even under its strict definition (high threshold). The fact that only 2.1% of the "IT naïve" users have not performed EUD activities under the low threshold shows that users have good EUD *Background*, i.e. they have experience with EUD and have adopted EUD practices in daily work.

2.1.3.4 Attitude Towards EUD for Task Management, Benefits and Drawbacks

After having introduced the notion of EUD and revealed common EUD practices, the questionnaire focused on the assessment of users' attitude towards EUD for task management and the perceived benefits and drawbacks from it. These issues were elaborated in Section 4 of the questionnaire. The questions and answers are given in Table 2.10. The analysis is based on 5-point Likert [Lik32] scale for the responses *Disagree strongly (-2), Disagree (-1), No opinion (0),*

Agree (1), Agree strongly (2). The results, except Q2, are based on 133 valid responses. The table also contains results of combined indicators.

Table 2.10: EUD attitude, benefits and drawbacks

Use	**M**	**SD**
Q1. I am already undertaking customizing or developing software related to task management (EUD for task management).	0.48	1.11
Q2. I am not yet involved in EUD for task management, but I intend to start doing this, given the right conditions and easy tools.	0.13	0.99
Combined Use = Max (Q1,Q2)	*0.83*	*0.86*
Self-Efficacy	**M**	**SD**
Q3. I believe I am able to successfully do EUD for task management, given just a software manual or a help facility.	0.90	0.80
Q4. I believe I am able to successfully do EUD for task management, provided I can call someone for help if I get stuck	0.94	0.75
Combined Self-Efficacy Scale (α=0.81)	0.92	0.71
Benefits	**M**	**SD**
The task management software and rules created by me can improve my effectiveness at work.	0.98	0.67
The task management software and rules created by me can provide me with better information about tasks and deadlines than the present software.	0.75	0.82
Knowing how to perform EUD may result in promotion and pay-rise.	0.04	0.96
My improved work effectiveness resulting from my EUD activities may result in promotion and pay-rise.	0.17	0.99
Knowing how to do EUD could be useful later, perhaps on other jobs or in a non-work context	0.89	0.74
Developing or customizing my software would be / is faster than waiting other people to do this for me	1.10	0.83
The software and task management rules defined by me are more likely to suit my needs than those developed by programmers.	0.83	0.80
I would find the software and task management rules created by my colleagues useful in my work.	0.63	0.65
I am /I expect I will be enjoying EUD because it introduces variety in the routine duties of my job.	0.71	0.82
Combined Benefits Scale (α=0.87)	*0.67*	*0.57*
Drawbacks	**M**	**SD**
If I was good at end user development, this may sidetrack me from my main career and result in a missed promotion because management would need my development skills where I am at present.	-0.32	0.96
If I made a mistake whilst developing my software, I will loose credibility and esteem in the office.	-0.38	0.97
If I leave the company my software will not be understood by my colleagues.	-0.08	0.94
The time I spent learning EUD was too long. / The time I spend learning EUD will be too long.	-0.22	0.77
The time I spend developing actual software and task management rules will be greater than the time saved by me and others once these are developed.	-0.16	0.92
The benefits of EUD are negated by the possibility that EUD may produce wrong results or crash.	-0.29	0.93
Combined Drawbacks Scale (α=0.75)	*-0.24*	*0.61*

A combined indicator of *Use* has been created to capture the combination of existing involvement in EUD for task management and intent to perform EUD. If a respondent has

answered positively Q1, that value is taken as an indicator of use, otherwise Q2 is shown and the answer regarding *Intent* is used as an indicator of *Use*. Computationally the bigger of the two values for Q1 and Q2 is taken. The results of both Q1 (133 responses) and Q2 (52 responses) are shown together with the combined indicator of *Use*.

The scales for *Self-Efficacy*, *Benefits* and *Drawbacks* are computed for each respondent as an average of the scores on the corresponding questions, after those have been subjected to reliability analysis within SPSS. The reliability analysis has produced very high Cronbach's α values (all >0.75) indicating that all individual questions are reliable indicators for their combined scale. The Cronbach's α values are reported in Table 2.10 in the row of each combined scale.

Following formula (2), a value for the *Expectation of economic outcome = benefit – drawbacks* is calculated for every respondent. The way this measure is calculated has been validated against the answers of a specific question testing the organizational expectation of economic outcome by the respondent: "Please consider the costs for your company arising out of any EUD-related errors in your task management software". The Spearman correlation test produced a correlation coefficient of 0.451, which is significant at the 0.01 level (2 tailed), thus providing support for the validity of the *Expectation of economic outcome* measure. The actual results from computing this measure indicate overall positive attitude to the balance between benefits and risks, with a mean value 0.92 on a -4 to +4 scale. This is significantly positive at any confidence limits up to 99.99% using one-sample t-test.

As reported in Table 2.11, testing for correlation between *Economic Outcome* and *Use*, and *Self Efficacy* and *Use* has provided some support to the underlying questionnaire model, showing that these variables tend to co-vary together. This partial validation is included here since it is focused on formula (2) which is a key theoretical aspect, underlying the questionnaire.

The results support the findings for prevalence of EUD, since a major part of the participants recognize that they are already undertaking EUD. The results further reveal that end users clearly recognize the potential to improve work by undertaking EUD. Their general attitude is rather positive, whereas drawbacks are rather met with disagreement.

Table 2.11: Correlation testing results

Var1	Var2	Pearson correlation	Sign (2-tailed)
Economic outcome	*Use*	0.322	<0.001
Self Efficacy	*Use*	0.537	<0.001

2.1.3.5 Supporting Actions

Supporting actions for EUD activities were given in Section 5 of the questionnaire. The results given in Table 2.12 demonstrate overall agreement with the effectiveness of the suggested set of actions.

Table 2.12: Supportive actions for EUD for task management

Supportive actions	**M**	**SD**
Hearing that someone I know has succeeded in EUD can stimulate me to try it myself	0.88	0.66
Hearing that someone has gained benefits from EUD can stimulate me to try this myself.	1.00	0.97
If my managers recognize the time spent on learning and doing EUD, I will be happy to engage in EUD or expand the scope of my EUD activities.	0.80	0.79
Attending a training course could help me to start using EUD or expand the scope of my EUD activities.	0.95	0.86
Introducing quality standards and mandatory testing in relation to user-developed software will decrease EUD risks.	0.82	0.90

2.1.4 Conclusions

In conclusion, the results of the survey support the premise that task management in small collaborative groups is a suitable ground for EUD activities. Indeed, the analysis of responses suggests that the median percentage of collaborative tasks is between 31% and 50%. Yet the overall complexity is not too high, with median and mode of between 3 and 5 people involved in the typical workplace task, involving average complexity of 5-8 operations by the individual. These findings suggest the use of semi-structured approaches to collaborative task management rather than conventional workflow systems which are tuned to the requirements of rigid task structures. The resulting generic requirements for user-centric task management support are summarized in Table 2.13.

Table 2.13: Basic requirements for user-centric task management systems – 'q' index denotes that a requirement results from the questionnaire

R_q 1	**Software environment**	For providing integrated task management support within a common end users' working environment, following software environments shall be considered:	
		R_q 1.1	***email as the primary environment for organizing and exchanging tasks (cf. Table 2.3)***
		R_q 1.2	***to-do lists is an extensively used environment for explicit task management (cf. Table 2.4)***
R_q 2	**Personal task management**	To efficiently support individual tasks a task management system shall support:	
		R_q 2.1	***due date setting and reminders (cf. Table 2.5)***
		R_q 2.2	***task order filtering (cf. Table 2.5)***
		R_q 2.3	***estimation of task processing time (cf. Table 2.5)***
		R_q 2.4	***integrated document management in tasks for supporting work with documents (cf. Table 2.2)***
R_q 3	**Collaboration support**	Due to the increased number of collaborative tasks, a user-centered task management system shall support collaborative task handling through:	
		R_q 3.1	***task delegation (cf. Table 2.5)***
		R_q 3.2	***task progress monitoring (cf. Table 2.5)***

The questionnaire results further show that users are already familiar with EUD techniques to an advanced level. Indeed some 75% of the "IT-naïve" respondents have declared familiarity with features such as setting filtering and forwarding rules, which are considered EUD activities. In the wider context, EUD practices were widely reported, with 50% of the "IT-naïve" respondents indicating that they have done at least five EUD-style activities. Generally, this suitability is reflected in the overall score related to the *Use* of EUD in the domain of task management. *Use* has a positive mean of 0.74 for the "IT-naïve" respondents.

This is in agreement with the positive results from the factors affecting users *Intent* to perform EUD (see (1) and (2)). *Use context* is considered positive in terms of suitability of the application domain for EUD (discussed above), and the mean for organizational support is 0.77 on a (-2 to 2) scale. *Individual judgment* indicates *Expectations of economic outcome* and *Self-efficacy*. The overall *Expectation of economic outcome* is positive, with a mean of 0.92 on a -4 to +4 scale. The combined result of benefits has a mean of 0.67 and the one on drawbacks has a mean of -0.24,

both on a (-2 to 2) scale. The combined results of *Self-efficacy* also produce a positive mean score of 0.92 on a (-2 to 2) scale. *Background prerequisites* are also positive, with only 2% of IT-naïve users not having done any operations regarded as EUD.

The positive *Intent* to perform EUD in the domain of task management is combined with positive indicators about *Resources* available for end users to perform EUD in terms of available time and technical support (see (3)). Respondents believe that IT support will be willing to help them with non-conventional queries, i.e. shown through a positive mean score of 0.22.

These results imply that task management systems can indeed strongly benefit from involving end users not only as executors but also as "developers". End users are qualified and committed for this extended role according to their general EUD experience level, estimation of benefits and drawbacks.

As a particular challenge remains to choose the most suitable EUD methods and to deliver appropriate *Technology* that takes maximum advantage of the users' preexisting EUD and task management experiences. EUD-enhanced *Technology* for business process management can benefit from preexisting user experience by leveraging end users' task management skills towards end-user driven process composition on enterprise level. Motivational usage barriers for the adoption of such technology can be minimized by delivering the expected benefits that have become manifest in the survey.

2.2 Assessment of User Problems

While the results from the questionnaire-based survey show that EUD in the task management domain can bring benefits to end users and is feasible, the empirical work discussed in the following identifies concrete problems that can be used to involve end users in process tailoring activities. This work comprises several phases of field studies which have been conducted independently from the questionnaire.

2.2.1 Studies of Informal Processes

The presented studies explored end user practices in informal business processes in three application partner companies. Throughout the dissertation company names are held anonymous for privacy reasons and the involved companies are referred to as: TXTL, SWVR and ASPL. An overview of the persons involved in the different study phases is given in Table 2.14.

Phase 1 consisted of semi-structured, face-to-face interviews with employees on managerial positions in all companies. The purpose was to gain an overview in organizational structures and common enterprise practices. Non-directed interviewing techniques [Kun03] were used. This phase revealed initial problem areas regarding informal business processes and helped to identify appropriate partners and target processes for the next phase. The interview guidelines for the start interviews are provided in Appendix A.

Phase 2 consisted of unstructured interviews aiming to identify generic end user work practices, common problem solving strategies and overall demand for transparency, knowledge exchange and guidance regarding ad-hoc business processes. Open-ended questions were asked, e.g. "What are your major pain-points in collaborative processes?". Refinement questions were asked depending on the interviewees' responses. All interviews were conducted at the site of the respective company in the familiar work place surrounding of the interviewees to preserve their context as far as possible. The participants were specifically selected for having knowledge-intensive tasks which required a lot of mutual agreements and informal communication.

Table 2.14: Users and their participation in the empirical study phases

Role	Company	Study Phase			
		1	2	3	4
Chief Officer	TXTL	x			
Chief Officer Assistant	TXTL		x		x
IT Department Lead	TXTL	x	x		x
Brand Manager	TXTL		x		
Marketing Employee	TXTL		x		
Production/Logistics Employee	TXTL		x		
Purchase Employee	TXTL		x		
IT Expert	TXTL		x	x	x
Sales Management Assistance	TXTL		x	x	
Chief Sales Officer	TXTL		x		x
Sales Employee 1	TXTL				x
Sales Employee 2	TXTL				x
Product Manager	SWVR	x	x		
Purchase Employee	SWVR		x		
IT Support Lead	SWVR		x		
Sales Support Employee	SWVR		x		
Marketing Employee	SWVR			x	
Controlling Employee	ASPL	x	x		
Customer Relations	ASPL		x	x	

Phase 3 consisted of semi-structured interviews, elaborating on concrete processes that were identified in phase 2. The purpose was to get detailed descriptions of informal business processes, including information about the used documents and involved persons. These descriptions could allow process analysis and identification of optimization possibilities. A Hierarchical Task Analysis (HTA) [Ann04] methodology was used. The analysis was based on interviews with domain experts that coordinated and managed the processes. The interviewees had to describe and sketch on paper the overall process flow and to provide artifacts (documents), used in these processes. The interview guideline for the process-focused interviews is provided in Appendix A. The interview data was later on transcribed as HTA diagrams [AS00, Ann04]. An example HTA diagram for a consignation sales process from the TXTL is provided in Appendix B.

Phase 4 of the user studies continued 2 months and focused on a user group, which was recruited for long term studies. Due to the heavy time-commitments and the extended requirements towards the participants (e.g. to have dependent, collaborative tasks within the scope of an end-to-end business process) only 6 users from the TXTL company could be recruited. This phase had the purpose to capture the work situation and individual practices before a concrete solution (prototype) was provided. It comprised contextual inquiries [BH98], which were shortly followed by clarifying, unstructured interviews elaborating on how the participants

use email or phone for managing their work, how complex are their tasks with respect to number of involved persons and used documents and how much they care about getting ways to shape, share and reuse efficiently process knowledge.

The following sections provide a brief overview of the involved companies and their software infrastructure and support to facilitate the understanding of the enterprise context, in which informal business processes take place. Then the findings from the field studies are summarized, which frame also the problem scope for informal process support.

2.2.2 TXTL Company

TXTL is a middle-sized manufacturing company with around 155 employees. The company is situated in Germany where it has 6 locations and additionally 2 subsidiaries – a dependent logistics center and an independent company, respectively with 60 and 20 employees. Raw material supply for the production, as well as a preliminary production cycle of semi-finished, goods are outsourced in Asia. Final production, marketing and distribution are performed in Germany. The company does not offer end-customer sales but works with retail vendors.

The company structure is relatively flat. There are partially department- and partially group leaders. There is a sales department leader which manages internal and external sales, production department lead, who manages production and logistics, independent financial accounting, human resources, purchase employees and brand managers. The business management and the marketing manager in the personal union constitute the company management. Additionally there are 2 different types of fieldworkers - classical and trend-sales, who focus on specific product areas. Within the company there is a tight meeting structure. Tight structures are preferred as far as possible to enable fast decisions and to keep the service paths for the customer short.

The budgeting in the company is made rigorously on annual basis. Monthly reports are delivered by the persons responsible for the budget in the different departments. There is a well defined business plan and a strict budget discipline is expected. In order to achieve the business plan objectives, active sales planning of the individual articles is performed.

The company has stable supplier relationships and works with some of the suppliers for more than 15 years. Common quality and communication standards with the suppliers have been built. The guiding philosophy thereby is to choose “not the cheapest but the best” suppliers. The relation to the business partners are maintained over telephone and email. No e-procurement is used. Adaptations of company practices and business processes are performed mainly upon external requirements from business partners or customers.

2.2.2.1 IT Infrastructure and Support

The software infrastructure in the company is maintained by an IT department with 5 employees. The IT department is rather business-driven, not technology-driven. The employees have degrees in business informatics. A driving principle is to have generalists rather than specialists. The advantage thereby is that IT employees have a better communication with the software users within the company and with external partners, and think more process-oriented. The basic responsibilities of the IT department are to elaborate on how new processes and requirements can be realized in the available software systems, to perform customization, maintenance and support of those systems, and in case of external consulting, to serve as an interface between external consultants and the company employees.

External consulting is acquired for special problems, e.g. with Windows 2k, SAP R/3 upgrade. Individual programming is not performed internally. Only small configurations are made. Key users accompany the external consultants during software installation and configuration activities in order to educate themselves. The key users can then partially maintain the systems without the need for external assistance.

The company implemented SAP R/3 system in 1999 because of year 2000 and EURO

changes. SAP was chosen for protection of the investment in long-term perspective. However the system is still considered too rigid and over-engineered. The SAP system is used by 104 employees, especially to support production and logistics. The implementation was oriented towards the anticipated target processes, and not towards supporting the current processes. A guiding principle was to achieve a possibly fast implementation and to adapt the system afterwards. The approach was result-oriented, i.e. the results determined also the further requirements for the system.

2.2.2.2 Process Support

Processes in TXTL run informally, mostly over email, but can include also face-to-face agreements and handing out of tasks in paper form. Microsoft Outlook is used as email client by all employees in the company. Reuse of templates is very common in processes. Templates are different documents in textual or tabular form, i.e. in Microsoft Word and Excel. These documents are stored on a structure of file folders on a central server, which contain personalized and department data. Documents are managed according to a numbering system. Thereby documents, such as e.g. customer contracts, are stored in the central file folders on the central storage and assigned numbers. These numbers are themselves maintained in Microsoft Excel tables on the central storage. The documents are referenced in processes merely through their corresponding number in emails, whereas no automated interface is available to fetch them from the central storage.

For process planning Microsoft Project is used in some divisions. Very wide spread is also the usage of MindManager [Min08]. It provides the desired features – hierarchical work breakdown and work assignment in areas of responsibility. This solution is considered somewhat "shirt-sleeved" but nevertheless sufficient in the most cases.

The process-focused interviews from phase 3 of the field studies in TXTL elaborated on a process for consignation sales. A consignment policy allows the customer, i.e. a retail vendor, to include products in their assortment for which they pay only after the products are sold. Products that are not sold can be returned to the delivering company. Through this, the risk of taking in too much or including new, "precarious"' products is eliminated. The delivering company has the advantage of being able to easily place new products. Also, competition can be squeezed out through the claiming of a relatively large portion of the sales area in a store. Smaller customers, especially retail vendors, mostly have no electronic ordering- or accounting system but "barely" access to email and Microsoft Office. Often, a business relationship between the company and the customer already exists. A consignment policy is mostly offered by the company to a given customer in order to boost new product groups. The consignation sales process was especially selected during the field studies as it involves different departments, extensive use of various document templates, and exchange of working documents. This process was further interesting because it had been established recently in the company, and was accompanied with increased need for knowledge reuse and guidance. A HTA of the process is provided in Appendix B.

2.2.3 SWVR Company

SWVR is a middle-sized company engaged in high-tech production and services. It has 6 locations in Germany and around 500 employees. Products are distributed through retail vendors and additionally through direct sales to end customers on subscription terms. For the direct customer sales the company provides also associated services through a large support call-center with about 70 employees.

The company structure is influenced through the high variety of offered products and services. The company consists of 12 departments with different functional and business domains, such as e.g. purchase, production, marketing, sales etc. The departments have managers that are reporting directly to the chief executive officer, who is also the company owner.

The company planning is performed in a top-down manner from the management. Resources are not planned because of the increased, constant work load – everybody is performing on their assigned tasks to keep things running in a timely fashion. No annual budgeting is made. The planning of extensions and customizations of the software infrastructure is triggered not by department leaders, but by key users of a given project team. The departments work project-driven whereby the manager of the company decides about the budgeting of the different projects.

Long-term, stable cooperation with external suppliers for the production and with the retail vendors exists. Changes in the internal policies and processes are mainly influenced by external market and legal conditions.

2.2.3.1 IT Infrastructure and Support

Due to the different business requirements in the different areas, the company has utilized a variety of software systems. The backbone of the company is in general a SAP R/3 ERP system. Additionally, MS Navision ERP system is used for the end-customer sales in parallel to the R/3 system because of the increased number of business partners (individual customers) and the high cost of the R/3 license resulting from that. The end-customer sales and related support activities are build-up as "a company in the company", whereas consolidation of the data in the R/3 and Navision systems is performed where necessary. Both ERP systems are complemented through a variety of proprietary modules for different departments, including production, marketing and sales. Some of this additional software has been developed within the company.

In the SAP software domain there is a large number of end users (40%-50%), "who only press buttons". About 30%-35% of the system users are domain experts, who understand the relevant concepts, i.e. what is happening in the background when certain operations are performed in the system. About 10% of the system users are key users. There is only one IT expert who knows to program. To be a key user in the SAP system is an official skill. A key user is process-responsible but does not have a disciplinary responsibility. Key users activities range from normatively shaping processes to supporting other employees by "telling which button should be pressed". Key users also communicate intensively among each other to synchronize on ongoing issues. There are employees who want to be a key user and also such employees that do not have other daily job but to act as key users.

The Navision software is used primarily in the call-center domain. There are only 20% full-time employees and the rest 80% are free lancers and short-term employees. Thereby there are very high differences between the skill set of SAP and Navision users. The latter are mostly students or persons who are not IT-educated or willing to engage with technology. There are only 2 key users for the Navision system. They participate in weekly meetings where solution proposals for certain issues are discussed with end users and middleware management. Smaller, informal meetings are also a common practice.

In SWVR it is considered very important that the end users approach and talk with the key users regarding system performance, necessary functionalities or other issues. The key users are familiar with the process context and can better evaluate the required functionality. Concepts for extending or customizing the system functionality are discussed and sketched by end users and key users, and realized by system administrators or free-lancers. The mode in which extensions and adaptations of the system are performed is generally reactive, i.e. addressing a specific need or request, and not proactive. Sometimes it can happen that concepts are proposed by IT, e.g. as a result from gained experience from conferences or exhibitions.

2.2.3.2 Process Support

Apart from the SAP and Navision systems, the software infrastructure at SWVR includes an exchange server. It is used with MS Office as standard office software. The usage of emails and tasks is widely spread and encouraged. Due to the lacking transparency resulting from increased

personal communication over telephone and email, Microsoft Project has been utilized for product publishing processes. Through this product managers can see the complete status of the ongoing activities. However, a lot of "conceptual" work is still executed informally, per email.

Further, the company has an internal portal, which is based on Active Server Pages and Java technology. It has been developed within the company. There is a documentation database, which can be accessed in the internal portal and provides self-services for the company employees. The hotline employees fetch Frequently Asked Questions (FAQ) documents from this database and send them to the end customers.

The process-focused interviews in SWVR elaborated on a process for the creation of a package (box) for a company product. The package of a product is an important part of the product marketing as it has the purpose to draw the attention of a potential customer, to present the product in the best way, and to clearly describe the benefits from using the product. This process was selected, because it involves different stakeholders from various departments such as marketing, product management, quality assurance, legal, customer relations, and graphics, who have increased need for coordination support. A number of documents, partially derived from templates, are exchanged and consolidated in clarification loops, to ensure that the contents of the package are correct with respect to wording, product characteristics and legal aspects, before the package printing is requested from an external printing house. Erroneous package content could result in delayed product delivery and losses for the company.

2.2.4 ASPL Company

ASPL is a middle-sized manufacturing company with about 120 employees. It is part of a larger holding encompassing 4 other manufacturing partners. The holding has emerged initially from the ASPL-company. The different companies have different internal structures and processes which are being aligned through ongoing efforts. The company is manufacturing materials and supplies for the production industry. The production cycle is determined by the number of items that need to be produced and delivered per day. Production is generally serial but make-to-order processes are not an exception. Clients are companies from all over Europe. The company is not involved in end-customer sales.

The company structure is relatively flat and is organized in different areas. There are product managers, quality assurance employees, sales, accounting, purchase, production, controlling, customer service and IT. Some of these areas are aligned in cross-functional departments which work in tight cooperation and share common processes. For example, there is an order management department, which is responsible for delivering a product from the customer order to the final delivery, and controls the overall production cycle. A material disposition department is responsible for production and materials whereas there is no classical separation between disposition, material management and sales. Price negotiations and master agreements for materials' supply are made in purchase. Disposition work is performed through the order management department. This aggregation of the responsibility areas has been made to streamline responsibility tracking and to avoid struggles between the different competencies.

The company planning is strongly focused on the actual products and customers. Planning activities start in the sales department, where key account managers evaluate production schedules provided by customers. These schedules describe the amount of products and the associated items lists which the customers will need to obtain from ASPL. After an initial sales plan is sketched, it is evaluated and produces the purchase and production plans. The capacity planning for the production determines also the overall investment planning – what needs to be optimized to achieve the expected performance. This involves also planning of IT investments, which are proposed by different department leaders. Such investments however mostly refer to small internal optimizations. Large scale investments need to be consolidated and approved on holding level as they can influence all partner companies. The whole budget is assembled from the planning rounds and clarified in several iterations. The companies in the holding are legally

independent and plan separately. The plans from the different companies are then brought together to frame the holding budget.

2.2.4.1 IT Infrastructure and Support

The IT department in ASPL consists only of one system administrator, who takes care only of the administration of the local network and users' desktop computers. The backbone of the software infrastructure is a SAP ERP system, which holds all productive data in the company. This system is outsourced and hosted through an external data center. The benefits are that the hosting price is manageable, and there are no difficulties to manage the system in terms of personnel. All areas work intensively with the SAP system especially production, sales, materials management, controlling and accounting.

The key user concept has been introduced with the SAP system and is widely spread. There are about 8 key users - one in each area. Key users work with the external SAP consultants and acquire knowledge about setting-up and customizing the system. The key users then multiply the knowledge and educate also the other employees. Key users are also identifying problems and communicating them to external consulting companies. There is a constant supervision of the SAP system. Assistance from a small consulting company is acquired to help maintaining the system. The consultants mostly look into processes by searching for erroneous behavior of the users and for optimization opportunities. Thereby not only employees from ASPL initiate system customizations but also the external consulting partner. Release changes of the ERP system are performed solely by the external SAP hosting center. The hosting center and the consulting company have customer hotlines for urgent support.

2.2.4.2 Process Support

Email communication is the main channel for work coordination in ASPL. Microsoft Outlook is used as the standard email client by all employees. Microsoft Project is used for long term planning in certain departments. However, most of the planning throughout the company is made in Microsoft Excel, which is considered extremely laborious and error-prone, but inevitable at this point. The exchange of standard documents with customers is realized over Electronic Data Interchange (EDI) over the SAP system.

The process-focused interviews in ASPL elaborated on a process for transfer of a product from prototyping to serial production. This process was of interest as it requires alignment of multiple departments and consolidation of various documents, partially produced from templates. The participants further have overlapping responsibility areas whereby peer support is often required to check that the process is running in a timely fashion and to avoid bottlenecks. The process is considered important in ASPL because it defines the final product characteristics and therewith also the product proposal for the customers.

2.2.5 Findings

This section summarizes the most important findings from the field studies, which are relevant for more than one of the involved companies and hence allow some generalization and abstraction from a specific case.

2.2.5.1 Prevalent Key-User Practice

The key user concept was spread in all companies that were involved in the empirical studies. This way of expertise management has been introduced with the SAP ERP systems. Nevertheless, this concept has been established as part of the companies' culture and focuses (business) domain expertise and technical skills in persons, who serve as interfaces between the end users of enterprise systems and the technical staff supporting these systems. Such users can efficiently play the role of "*gardeners*" [GN92] for EUD activities in enterprises.

2.2.5.2 Overlapping User Roles

In the process descriptions that were elaborated in phase 3 of the field studies, certain business roles were used to sketch the different expertise areas and work assignments. However, in some cases these roles overlapped in terms of personnel. For example, while describing the consignation process, interviewees talked about financial accounting. The latter was however performed by a person, who had a broader expertise scope in the sales area. This person was responsible also for activities like budget planning and was further acting as a sales manager for specific sales areas. These overlapping business roles were a common practice also in the other SMEs, which could be explained through the limited human resources. Furthermore, task assignments in informal processes such as the consignation sales in TXTL were not strict. For example, if a consignation of an important customer was handled, sales management was directly involved in many cases instead of its assistance or sales support. This flexibility was considered as an intrinsic part of the company's culture that allows the company to respond more adequately to changes and problems in the business process.

2.2.5.3 Email as the Primary Organizational Interface

Email was used as the primary channel for task and information exchange in all companies. This finding is largely supported also in related research [BDHS03, BDH+05] which promotes email as the main organizational interface between knowledge-workers. In all companies negotiations with raw material suppliers and customers were performed over email. Email further facilitated responsibility tracking and distribution. For example in case of escalations higher management was included with CC and the escalation evolution was traceable over the email threads. Email was considered also a mean to create personal attitude and to include informal connotations in formal processes. For example, in the customer support department of SWVP each customer received a personally written email response, although the major part of the problem resolution procedure was automated and answer templates were available. Such practices were part of the companies' culture and were not subject to change.

2.2.5.4 Personal Task Management

Nine of the interviewees, especially those with senior functions and higher responsibility, maintained to-do items in a Microsoft Outlook task list. The major benefit from managing to-do items was that these provide automated reminders for upcoming tasks when the users start their email client, i.e. Outlook. Rescheduling and delegation of such to-do items was also used. For example, the sales manager in TXTL coordinated the work of the sales and sales management assistance employees through distribution of Outlook tasks. Task items in to-do lists appeared to be a commonly used explicit representation of end user tasks in informal processes, which agrees also with the findings from the questionnaire-based study. These findings agree also with related work on usage of to-do items for personal task management [BDG+04].

2.2.5.5 Lacking Motivation for Use of Formal Systems

Users' working conditions and time constraints strongly influenced their attitude towards maintaining data in software systems. Explicit planning was performed only for long-term projects, i.e. through product managers and other employees that needed to coordinate the timely delivery of the overall project result. Therefore Microsoft Project was used only by small groups of people in the different companies. Most of the users, who worked on tactical tasks with weekly or monthly scope, considered the utilization of more-formal systems as rather restrictive. For example, a marketing employee in SWVP reported that Microsoft Project was gradually put out of use in his and other departments, as they did not have such demand on planning, and the use of this tool did not compensate their efforts. As a result of this lacking motivation for use of formal

systems, work coordination in informal processes such as the product package creation in SWVR was not supported through an overview of the evolving collaborative tasks.

2.2.5.6 Uncontrolled Document Management

All processes that were investigated in phase 3 of the field studies involved a lot of manual tasks of different stakeholders that made an extensive use of different document templates, updated or produced new documents. In the consignment process templates were used e.g. for the consignation contract, for inventory lists and stock reporting. In the process for transfer of prototypes to serial production in ASPL, templates were used for quality check reports and manufacturing instructions. During running processes documents were exchanged over email and edited locally in personal workspaces in a rather asynchronous manner, i.e. without versioning or access policy.

2.2.5.7 Process Guidelines vs. Implicit Process Knowledge

A further important finding is that reoccurring processes exist and are documented to a different level of details. For example, the process for consignation sales in TXTL has been established over time. Detailed guidelines for this process have been created in the TXTL company. These guidelines were provided as textual documents, which described the areas of responsibility and the tasks that should be performed by the different departments. Even a MindManager [Min08] mind-map was available with an overview of the work distribution. In ASPL a guideline for the process for product transfer from prototyping to production was also available as a Microsoft Excel table. However, the provided guidelines only roughly framed the processes. The actual process flow was determined solely though the ad-hoc coordination of the individual activities and through information exchange over email. Personal email and file folders were the final storage where information about executed processes was kept, hence opposing the implicit knowledge of the actual process execution to the provided process execution guidelines.

2.2.6 Problems

This section focuses on the problems that end users face during the management of informal business processes. These problems complement the discussed findings for current practices in informal processes, and reveal entry points for introducing process tailoring to end users. The findings and the discussed problems motivate the requirements for end-user driven business process composition presented later on.

2.2.6.1 Lacking Transparency

As users mostly exchanged process related information over email, there was no overview and no feedback about the status of evolving collaborative activities. This lack of transparency was especially critical for employees who were responsible for managing and coordinating processes. A controlling employee from ASPL exemplified that clearly for the area of price estimations for new products (prototypes):

"For example you receive a product sample and you are told: "Ok, now give me a price for it." … first it [Excel spreadsheet for calculation] is set up from a sales employee, then a cooperation with product development starts, then it [spreadsheet] goes to a calculator, then it goes back to a sales employee, then something else is modified, a new component is inserted, then changes from customer [requirements] are entered, so that it [spreadsheet] floats, it is dynamic, it evolves. You can lose the overview relatively fast, it can relatively fast happen that someone that you needed to inform and involve was simply forgotten … or someone has an information from release 4 and the other from release 10 and then someone enters something wrong because it was the pre-pre-pre-state of the data … people are generally instructed that the everybody knows what they are doing

and that everybody is giving the others the right information to the right time … And this could work essentially better … And it can become dramatic if due to a wrong calculation or calculation state some [wrong] product proposals are made to customers"

Hence, for coordinating cooperative processes transparency of involved persons, task flow and document flow is necessary similarly to conventional workflows. A benefit from such transparency, which was also mentioned by a sales employee in TXTL, is that having other persons involved as peers can help to detect discrepancies, resulting e.g. from the fact that important stakeholders are not involved, and to correct these discrepancies before time losses or additional costs have occurred. Involving peers could further help to identify optimization possibilities.

2.2.6.2 No Structured Storage and Retrieval of Process Knowledge

While process-related data was distributed in personal email and file folders, and managed on central file shares, users had developed individual strategies for sorting this information. For example emails were often sorted in personal file folders named by products and customers. However, users were not able to predict how their current "sorting" practice will scale over time. Increasing data amount made the storage of information in email and file folders less efficient due to the increased search efforts. An IT employee from TXTL exemplified the problem:

"If I need to see, how or why something happened I look in my email folders. […] If I cannot find an appropriate email, I browse to our transaction tables on the share [MS Excel tables, where all business transactions are maintained]. There I look for a transaction by approximately knowing the date when it happened and what was it about [customer, product]. I copy the transaction number and then I go and search for this number in the emails to find the appropriate message or mail thread."

Hence, finding or assembling data that is relevant to earlier activities can often involve a lot of manual operations and access to different, distributed information sources. The problem of increased time consumption for searching information in email folders is raised in related literature [BDHS03]. Such practices can degrade individual and group performance.

2.2.6.3 Lacking Exchange of Up-to-Date Process Knowledge

All participants in the preliminary studies were confident that they know how tasks have to be processed. However the need for an explicit, global representation of process knowledge was clearly perceived in all companies, especially when certain domain experts were not available to provide assistance on time critical activities. Such global documentation was available to a different degree of details in a static, textual form in the different companies. For example in ASPL and SWVR guidelines were provided on internal portals and in text-based documents in shared folders. The most comprehensive knowledge management strategy was developed at TXTL. This strategy is evident from the comments of two employees with managerial functions in the following interview excerpts:

Interviewer: "Is that something that could be supported in some way that know-how that grows in the company, that is there, does not exist only in the heads of individual persons and as soon as they go it disappears with them? Is there an informal way to manage knowledge?"

Interviewee 1:"Such does not exist. There is an embedded protection … we have structured the different areas, namely, the people must describe their tasks, respectively their main and sub-tasks. The main tasks are the elementary tasks that are necessary to keep the company alive. And

each employee has main tasks. You could say also they are divided into priorities A, B and C or something like that. The tasks should be described respectively through a so called purpose description and execution description and then the means and the measures must be described – for accomplishing a task I need a pen and paper, or I need a computer or I need a scanner etc., and then a task has a goal. And we say if a task has no goal, if the employee cannot write down the goal of the task then they do not need to do that, they can throw the task away. And this brings naturally also more discipline. And how do we capture knowledge know-how. Very simply - at least two persons should be able to accomplish A-tasks [priority A] until the last detail, this is obligatory… It should be like a cooking recipe. Should someone lay down in plaster cast in the hospital tomorrow, the colleague should be able to open a folder and to actually know how this [task] functions. How do I do an export to Italy, what do I need for that? This and this form, then I must go to my computer and do this and that."

While the knowledge management strategy described in the above interview excerpt seems rather comprehensive, it actually turned out to describe the ideal case. It became apparent that such static, text-based documentation faces barriers when it comes to maintaining it and keeping it up-to-date. This was clearly stated later on in the interview:

Interviewee 1:"But there you work always against resistance. These organizational things that I have just described they are not loved by the people. Nobody at all does that gladly. You have to always ask people, have you documented your tasks, have your done the task description, is the execution exactly described, if someone has to do that [task] at some point, does that function, does that work. You have to always make pressure."

Interviewee 2:"They have to maintain this [task descriptions]. In the way that he [Interviewee 1] has described it, we actually do not have it currently. There are a lot of people that have to rework it [task descriptions]"

Eventually, the up-to-date process knowledge was held in the local email and file folders of various employees, who had to perform laborious, additional work to feed it into the global documentation. Updates were often avoided due to the increased overhead. This resulted in inconsistent or unreliable documentation and finally in loss of task and process knowledge.

2.2.6.4 Disjunction Between Best-Practices and Running Processes

Existing recommendations for globally relevant work practices in the form of text documents did not provide the possibility to follow evolving user tasks with respect to the provided guidelines. Users simply read the documents and executed the described steps in their highly variable working environment, where no comparison to what extent the described practice is being followed, or an explanation, why is it not being followed, is available. A short excerpt from an interview with a sales employee in TXTL exemplifies that:

"Well yes we have all the sales processes documented. [Pause] Every workplace has its [process] descriptions. These are locally available."
Interviewer: "In textual form?"
"Exactly"
Interviewee: "Are there cases when it is written this and this step should be performed but they are not"
"[laughing, slightly awkward] yes, we have that, we look - aha, that we have done earlier, and why are we actually not doing it any more? … yes, we definitely have that"

Such discrepancies often resulted from the fact, that maintaining detailed documentation for each specific case was laborious. If a case was considered exceptional, people did not bother to provide a guideline how exactly such exceptions need to be handled because these would normally not appear any more. In long-term aspect, the discrepancies between running processes and best-practices could lead to repetition of erroneous behavior.

2.2.6.5 Inability to Trace Evolving Best-Practices

Changes in certain work practices were often expected due to the dynamic business environment. However, the information describing these practices was spread over email and text documents in the workspaces of different users. Therefore no visibility was available of how exactly these practices have changed, and why this has happened. The respective knowledge was stuck in specific domains and could be extracted only with extra effort by contacting the domain experts. A senior sales employee from TXTL commented:

"There are common, shared documents but everyone takes these, makes local copies and local notes in them … everybody finds it useful to have a place to look up how something should be done, but everybody has their own interpretation"

As a result personal best-practices refined the generic guidelines or could even change these over time if an exception becomes a rule like in the case of consignation sales. But this transformation was not traceable as all the information related to a specific business case was spread throughout the individual workspaces.

2.3 Requirements for End-User Driven Business Process Composition

The empirical work resulted in a set of generic requirements for end-user driven business process composition. These are summarized in the following sections by considering input from both - the questionnaire-based survey and the field studies. Generic requirement definitions are provided in the following rather than concrete requirement specifications. These generic definitions are referenced throughout the dissertation to motivate the elaborated concepts.

R1: Personal Task Management in Light-Weight To-Do Lists

The field studies pointed at personal to-do items as the only explicit representation of the individual tasks in informal processes that allows users to manage, delegate and monitor tasks. These findings agree with the results from the questionnaire, which point at an extensive use of to-do items for management of individual and cooperative activities (cf. Table 2.13, R_q 1.2). The results from the questionnaire-based study have further identified detailed feature requirements for the management of to-do items (cf. Table 2.13, R_q 2.1, 2.2 and 2.3).

Furthermore, as to-do lists explicate processes on individual level, these can enable structured storage and retrieval of process knowledge (cf. Section 2.2.6.2). For this purpose, some advanced structuring mechanisms such as hierarchical task decomposition should be considered. A system for end-user driven business process composition should hence enable end users to specify, refine and manage individual tasks in personal, hierarchical to-do lists, which are also the natural users' environment for personal task management.

R2: Email-Based Person-to-Person Communication for Exchange of Tasks and Deliverables

The field studies and the questionnaire (cf. Table 2.13, R_q 1.1) clearly leverage email as the mostly used tool for coordinating work and exchanging information in informal business processes. Email hence emerges as a natural integration environment for user-centric support in

informal business processes. A system for end-user driven business process composition should hence enable end users to exchange tasks and task-related information over email. In this respect also the transition between to-do items and emails and vice versa should be considered, i.e. a system should enable export of emails as to-do items and delegation of to-do items per email (cf. Table 2.13, R_q 3.1).

R3: Transparency of Tasks, Documents and Persons in Informal Processes

Transparency in ad-hoc business processes was clearly stated as an issue during the field studies. The importance of task progress monitoring is amplified also through the results from the questionnaire-based survey (cf. Table 2.13, R_q 3.2). Hence, a system for end-user driven business process composition can deliver added value to the end users by providing them with an end-to-end overview of evolving, collaborative tasks including task progress information, document flow and involved persons. Document flow is especially relevant with respect to the increased document use in tasks, which manifested in the field studies and questionnaire results (cf. Table 2.13, R_q 2.4). Considering (R1) and (R2), the system should enable transparency in evolving ad-hoc business processes by integrating the individual to-do lists of different process participants based on the task delegation over email, and by additionally providing relevant context information of task instances such as attached documents, involved persons, and dialog flow for the transfer of tasks and deliverables.

R4: Exchange, Adaptation and Reuse of Process Knowledge

The need to exchange and reuse up-to-date process knowledge requires that generated real-life process data is globally accessible in the enterprise, so that this data can be adapted and reused in similar cases. To achieve this, a system for end-user driven business process composition should enable publishing of individual tasks and overall enterprise processes together with all contained context information about used documents and involved persons to personal and enterprise repositories, respectively for individual and group reuse. Publishing and reuse can lead to the recognition of best-practices for the handling of recurring business cases. Adaptation of best-practice templates should be further enabled to address deviations in different cases of reuse. Thus captured best-practices should provide guidance and not strict rules for the execution of informal business processes.

R5: Interconnection of Best-Practices and Running Processes

The field studies showed that there are difficulties in tracing how guidelines are being followed in running processes. A failure to detect deviations from provided guidelines in running processes can lead to the repetition of erroneous behavior. Therefore, the system for end-user driven business process composition should enable structured comparison between best-practice templates and running processes. The comparison would allow users to estimate the deviations and to evaluate, to what extent the guideline is being followed or why deviations have occurred.

R6: Tracing of Best-Practice Evolution

Best-practices can change on personal and organizational level to take into account optimization possibilities, or to adapt to changing internal or external business conditions. Thereby it is not only important to know what is the current best-practice, but also to be able to see what were previous practices and why changes were introduced. The system should hence enable interrelation and structured comparison between best-practice guidelines for similar cases.

R7: Process Automation

The consignation sales process that was explored during the field studies in TXTL has been established over time and evolved from a rather informal process to a well-documented

procedure. A similar tendency was observed also in the process for prototype transfer to serial production in ASPL. Although tasks in such processes were still managed through ad-hoc coordination over email, the rigidity with which these tasks needed to be documented and performed correlates to structured workflows. The need to establish such processes in a more-formal manner can be resolved through process automation on a workflow engine. However, starting formal process modeling from scratch can lead to inconsistencies, e.g. through wrong interpretation of textual guidelines or user statements from interviews. Furthermore, process modeling from scratch can be very time-consuming. Time and costs for formal process modeling can be reduced, if previous knowledge of the process is utilized, which is real-life compliant and established by the end users themselves. Hence, the system should be able to formalize user-defined process models, emerging from personal task management and email exchange for task delegation, towards the automation of rigidly recurring processes on workflow engines.

2.4 Summary

This chapter has presented the empirical foundations for the conceptual work in this dissertation. The methodology and findings from a questionnaire-based survey and from filed studies at industrial companies have been presented.

The questionnaire-based study assesses existing end-user work practices and perceptions of benefits and risks regarding EUD in the domain of task management. This domain has been chosen because of its particular relevance for bridging the individual perspective on personal task management with the enterprise BPM perspective [RRMvdA05]. The results from the questionnaire-based survey show a high prevalence of EUD practices. The results further reveal positive attitude of end users towards the utilization of EUD methods for optimizing individual performance through user-tailored task management.

The results from the questionnaire-based survey are complemented through empirical findings from field studies at industrial partner companies. The field studies provide an assessment of user problems in informal processes. The discussed problem areas can be addressed to involve end users in process tailoring by delivering added value to them in the day-to-day work.

Based on the findings from the empirical work, a set of requirements for a system for end-user driven business process composition have been elicited. These are considered during the state of the art analysis on user-centric process support to exemplify the benefits and shortcomings of existing approaches and to motivate the original contributions of the dissertation.

CHAPTER 3: Business Process Composition - State of the Art

This chapter provides a state of the art analysis on business process composition by considering the initial challenges for Business Process Management (BPM) systems and the findings from the preliminary empirical work. The focus is set especially on the applicability of related approaches for process composition by technically non-skilled end users. The primarily addressed research area of End-User Development (EUD) is explored to identify fundamental concepts for the creation and modification of software artifacts by end users, and to assess different approaches for achieving end-user driven process composition. These observations are complemented with a discussion on user-centric process support for different process types with respect to their structure and predictability. Composition of structured processes is discussed in the context of workflow management systems, BPM systems and Process-Aware Information Systems. Composition of semi-structured and unstructured processes is discussed through input from different research areas such as Computer-Supported Cooperative Work (CSCW) and knowledge management. The discussion of the approaches for user-centric process support is underpinned through the identified fundamental EUD concepts and the requirements from the empirical work.

3.1 End-User Development

The basic premise of the dissertation is that end-user driven business process composition can be enabled through EUD. Therefore EUD is the primary research field addressed. EUD integrates different threads of discussion from human-computer interaction, software engineering, CSCW and artificial intelligence, and different concepts and related terms that have emerged over the last two decades, such as *"end user programming"* [Cyp93], *"end user computing"* [BB93] and *"end user software engineering"* [BCR04]. Hence, EUD aims to provide a holistic view on the adaptation of software systems by end users and on user-centric system design, by exploiting synergies between related concepts and research fields.

3.1.1 Fundamental Concepts

The EUD concepts discussed in the following provide the scientific foundation for the theoretical work on end-user driven business process composition. These concepts address generic problem areas with respect to creation and adaptation of software artifacts by end users. While some of the concepts are introduced in related literature with respect to system tailoring, in the following these are translated for the domain of business process composition. In this domain the concepts are equally relevant, considering process models as non-trivial software artifacts that are developed and adapted by end users.

3.1.1.1 Gentle Slope of Complexity

[MCLM90] motivates the need for user-tailorable software systems through the fact that it is impossible to design systems that are *"appropriate for all users in all situations"*. Tailoring by end users is seen as a possibility to adapt software to match the individual end users' work practices and to increase users' performance. The idea to build systems which end users can *"design-during-use"* is intrinsic for EUD and can be found in a large body of research [MM00, FGY+04, LPKW06, SDW08]. However, [MCLM90] raises the issue that system tailoring by end users *"requires not only systems which can be tailored but a culture with which users feel in control of the system and in which tailoring is the norm"*. In this respect the latter study considers different user types, and exemplifies the need to ensure a *"gentle slope of complexity"* for end users engaging in tailoring activities, i.e. by enabling end users to evolve a tailoring culture,

without confronting them with technical environments that exceed their IT-expertise and discourage them from creating or modifying software artifacts. Through this also a transition between the different user types (cf. Section 1.5.1) can be enabled, allowing end users (workers) to climb the *"tailorability mountain"* [MCLM90] and to eventually become local developers (tinkerers) by extending their EUD skills. In the context of business process composition, the gentle slope of complexity means that end users should not be confronted with upfront process modeling environments and notations that exceed their technical skills and process understanding. This relates to the findings from the empirical work, showing lacking motivation for the usage of formal systems on a daily basis (cf. Section 2.2.5.5). Therefore the thesis considers a gentle slope of complexity as a basic meta-approach for realizing end-user driven process composition.

3.1.1.2 Reduced Expertise Tension

While the concept of a gentle slope of complexity is rather focused on the technical complexity of a software system, [Ber04] raises the importance of *"reduced expertise tension"* also with respect to the job-related domain knowledge. That is to say, EUD support needs to assist end users not only in adapting software artifacts without requiring extensive system-related knowledge, but also in tailoring software artifacts according to the right business domain expertise. For the area of process modeling this means that user-defined process models need to be consistent not only according to a given process modeling notation, but also that these models need to incorporate correct process information from business point of view. The concept of reducing the expertise tension relates to findings from the empirical studies. On the one hand, the need for transparency in evolving business processes (cf. Section 2.2.6.1) points that individual process tailoring activities in running processes should be supported through detailed information about the current business context in which the tailoring takes place. This information can help to ensure that no discrepancies are introduced during the tailoring activities. On the other hand, the need to enable exchange and reuse of up-to-date process knowledge (cf. Section 2.2.6.3) points that process tailoring should be supported through guidance based on previous experience for similar cases. Hence, the reduced expertise tension is considered as a meta-approach for ensuring composition of process models that are consistent with respect to a given business context.

3.1.1.3 Seeding, Evolutionary Growth and Reseeding (SER)

It can be hardly assumed that end users will be able to compose detailed and consistent process definitions in a straightforward manner, not only because of the different level of technical skills and business domain expertise of end users, but also because of their different attitude towards maintaining process data. Hence, user-defined processes can be considered as underspecified, whereas adaptation and extension of these processes can be required in a concrete execution context. Related literature explores the need for enabling enhanced system adaptation in the context of use by proposing the "meta-design" approach [FGY+04]. According to the latter study *"Meta-design characterizes objectives, techniques and processes for creating new media and environments allowing "owners of problems" (that is, end users) to act as designers"* [FGY+04]. For supporting meta-design [FGY+04] proposes the *"seeding, evolutionary growth, and reseeding (SER)"* process model. This model postulates that evolving software systems should continually alternate between periods of activity, unplanned evolution, and periods of deliberate (re)structuring and enhancement. In the domain of business process composition, this translates into a need to enable publishing, adaptation and reuse of user-defined process models towards their iterative refinement and complementation. Thereby adaptation should be enabled at use time, i.e. while the processes are executed, and at design time, when process outcomes are evaluated and captured process models are optimized. The SER concept strongly correlates to the findings from the empirical work about the need to enable exchange of up-to-date process knowledge (cf. Section 2.2.6.3), to interconnect process guidelines with running processes (cf.

Section 2.2.6.4) and to trace the evolution of best-practice guidelines in different cases (cf. Section 2.2.6.5). Hence, SER is considered as a concept that needs to be supported by environments for end-user driven business process composition.

3.1.1.4 Process Tailoring as Collaboration

Meta-design postulates the idea of shifting fine-grained system adaptation from formal system development to the end users' working context. This brings forth issues about collaboration and knowledge exchange between end users and developers. Such collaboration can bridge the understanding of both parties about the tailored software artifacts and about the associated requirements. The need to support effective communication and shared understanding in long-term tailoring is discussed in [MM00]. The latter study discusses among others process tailoring as collaboration in the context of workflow modeling. [MM00] shows the need to enable shared understanding of process models between end-user tailors and software developers. This closely relates to the need for "*increased business collaboration in process modeling*" that is perceived in the industry [For06], and to the second major challenge for BPM systems addressed in this thesis (cf. Section 1.2.2). However, [MM00] shows that process tailoring through visual notations requires "*both domain expertise and advanced skills in computer use*". Therefore such tailoring is inappropriate for technically inexperienced end users. Hence, a particular challenge for process tailoring as collaboration is to involve end users without technical skills in process composition, and to provide a shared context between user-defined and formal process models where end users, process designers and developers can work collaboratively on the consolidation and adaptation of process models. The thesis addresses the challenge of providing such a shared context and considers process tailoring as collaboration as a basic meta-approach for enabling end-user driven business process composition.

3.1.1.5 Fundamental EUD Concepts and Related Requirements

Several fundamental concepts have been identified for enabling end-user driven business process composition:

- *gentle slope of complexity [MCLM90]*
- *reduced expertise tension [Ber04]*
- *seeding, evolutionary growth and reseeding (SER) [FGY+04]*
- *tailoring as collaboration [MM00]*

These concepts determine also major design goals for user-centric process composition environments. Furthermore, the identified fundamental EUD concepts clearly relate to the requirements from the empirical studies. The relationships are given in Table 3.1 and discussed in the following.

For ensuring a gentle slope of complexity, end-user driven business process composition approaches need to leverage end users' experiences with standard applications for task management and collaboration. This can be achieved through integrating the process composition environment in light-weight to-do list applications for personal task management (R1), and in email environments for person-to-person communication for task delegation (R2).

Reduced expertise tension can be achieved through providing transparency in evolving collaborative tasks. Transparency can enable end users to tailor evolving processes at use time by estimating possible approaching deadlines and escalations, and viewing relevant flow of tasks in their area of responsibility, i.e. according to the actual business context (R3). A reduced expertise tension can be further supported through exchange, adaptation and reuse of process knowledge (R4) which enables users to receive guidance based on personal and organizational best-practices.

Exchange, adaptation and reuse of process knowledge in the form of individual task hierarchies and overall enterprise processes (R4) enable SER of business process definitions for their complementation and refinement. If best-practice reuse is not supported through structured

comparison between the adopted best-practices and the running processes that use them, guidance in running activities can be hindered and previous erroneous practice can be repeated. Hence, SER can be facilitated through interconnection of best-practices and running processes (R5). Furthermore, SER without maintaining relationships between different variances of a best-practice can hide varying business requirements and lead to inappropriate guidance. Therefore SER in the context of end-user driven business process composition can be enhanced through tracing of best-practice evolutions (R6).

Process automation (R7) can be achieved by using a conventional BPM or workflow management system. However, these systems need an explicit, formal process model to operate [vdAvH02, vdAHW03]. The elaboration of formal process models is performed by business technology experts, i.e. process designers and developers. On the other hand, the requirements towards the process design and automation come from business users and domain experts. The latter are generally unfamiliar with the formal and technical aspects of process modeling so their requirements need to be correctly understood and implemented by the business technology experts. Thus, a broad space for misinterpretation of requirements exists in workflow projects and can lead to their failure [Her00]. The need arises to facilitate process modeling towards automation of recurring processes by bridging the business and technology perspectives on process models and enabling process tailoring as collaboration between business users, process designers and developers [MM00]. One way of bridging the different perspectives on business processes is by deriving formal process models from user-defined process models. Thereby, end-user driven process composition environments can allow process designers and developers to extend user-defined process models in a shared context, integrating real-life and formal process representations (see also [MM00]). A further consideration for enabling process tailoring as collaboration is to extend the SER capabilities towards refinement of formal process models by end users at runtime. User-defined extensions at runtime can lead to adaptable workflows and address the third major challenge for BPM systems (cf. Section 1.2.3).

Table 3.1: Overview of fundamental EUD concepts and related requirements

Gentle slope of complexity	(R1)	Personal Task Management in Light-Weight To-Do Lists
	(R2)	Email-Based Person-to-Person Communication for Exchange of Tasks and Deliverables
Reduced expertise tension	(R3)	Transparency of Tasks, Documents and Persons in Informal Processes
	(R4)	Exchange, Adaptation and Reuse of Process Knowledge
SER	(R5)	Interconnection of Best-Practices and Running Processes
	(R6)	Tracing of Best-Practice Evolution
Process tailoring as collaboration	(R7)	Process Automation

3.1.2 EUD Approaches

The discussed fundamental concepts provide basic meta-approaches and reveal major design goals for realizing end-user driven business process composition. A discussion of concrete EUD approaches that enable end users to construct and adapt software artifacts is further needed, to assess possibilities for the actual composition of business process models by end users.

Related literature identifies two generic types of EUD activities from a user-centered design

perspective: (i) *parameterization and customization* and (ii) *program creation and modification* [LPKW06]. A similar classification of EUD activities can be found also in [CFMP06]. Activities from the first type allow users to change already available behavior or representations within applications. As customization activities are considered for example the change of the look-and-feel, the application of different themes or the adjustment of buttons in toolbars as this can be performed e.g. in Microsoft Office applications. Parameterization can additionally adjust dynamic behavior. An example activity is the setting of email filtering rules. Activities from the second generic EUD activity type imply the creation of software artifacts from scratch or internal modifications of such artifacts. Such activities are considered more relevant with respect to EUD. Nevertheless, customization and parameterization can enable smooth involvement of end users in more complex tailoring, i.e. users can start from customization and parameterization and move towards more complex creation and modification of software artifacts.

While [LPKW06, CFMP06] provide general classification of EUD activities, for enabling end-user driven business process composition it is important to evaluate what user types are addressed and what system adaptations are enabled by the different EUD approaches, i.e. to consider EUD approaches from user-centric and formal systems' design perspective. A survey of user perspectives on different EUD approaches can be found in [Bla06], where EUD approaches are discussed in the context of different user groups such as application users, technical users, educational users and business users. Thereby the latter study does not consider a generalized classification of the addressed user types in terms of required IT skills, such as end users, local developers or programmers (cf. Section 1.5.1). A discussion of EUD approaches from systems' design perspective is provided in [SDW08]. The latter study classifies EUD approaches based on their complexity and adaptation power. Complexity is expressed through the addressed user type: *"non-programmer"*, *"local developer"*, and *"programmer"*, where the first type corresponds to an inexperienced end user (cf. Section 1.5.1). The adaptation power on the other hand is divided in three different categories *"customization"*, *"integration"*, and *"extension"*. The *customization* category shares the previously introduced interpretation from [LPKW06] and is clearly unable to address process composition by end users due to the restricted adaptation power. The *integration* category represents *"approaches, which allow changing the internal design of applications by using some kind of model"* [SDW08]. This category particularly matches the goals of end-user driven business process composition, i.e. to enable end users to create process models from scratch and to modify these models in the context of use, during process execution.

The different classifications of EUD approaches provide some implications about possibilities to enable end-user driven business process composition. However, a detailed discussion is needed to fully assess the different EUD approaches with respect to the target user group for business process composition addressed in the thesis (technically inexperienced end users), and with respect to the identified fundamental EUD concepts and requirements from the empirical studies.

3.1.2.1 Programming by Example (PBE)

Programming by Example (PBE), also called programming by demonstration, [Cyp93, Lie01] enables end users to capture system behavior while acting in a software system as usual, and to provide this captured behavior to the system for re-execution. The most widely spread PBE practice is for example the recording of macros in Microsoft Office applications for automation of recurring tasks. Due to the increased unobtrusiveness and tight integration in the actual end users' working environment, this approach closely relates to the concept of *"direct manipulation"*, where *"the user is not required to interact in the interface domain of computational abstraction, but works directly with the data that interests him or her"* [Bla06]. The power of this approach is hence that users are not required to have technical or programming knowledge, or to engage with programming notations. The system generates the program for them by generalizing their actual, captured behavior. [RI06] further exemplifies how PBE facilitates programming by practically removing the possibility for syntactical errors as opposed to textual programming. The latter

study refers to a game programming example, which uses PBE through graphical rewrite rules. [Let06] further leverages PBE as a possibility to perform *"programming with the user interface"* which *"ideally means programming at the task level, which is more familiar to the end-user"*. The importance of PBE is additionally leveraged in [DSB06] where PBE is seen as one of the few techniques which can nicely combine usability and specification issues and which successfully addresses the lacking technical background, motivation and time of end users to engage with traditional programming approaches.

One weakness of PBE is that it may be difficult for an end user to get an insight into the captured execution example and to modify it if they have to learn a complex programming notation or machine language [SCT01]. Thus, it may become difficult for an end user to reuse and apply a captured example in a different context which requires a slightly different procedure or different information artifacts [Bla06]. These issues are addressed in related research by accompanying PBE with generation of visual or textual descriptions of the captured system operations, which have different degree of formality and allow end users to modify them at a later stage [Bla06].

In a nutshell, PBE is a promising technique for enabling end-user driven business process composition as it enables integrated support in a common end user working environment. PBE further relies only on direct manipulation of system objects and does not require knowledge of a given formal process modeling notation. Hence, it is able to address end users without any process modeling or programming skills.

3.1.2.2 Visual Languages

Visual languages can assist expert users to model their problem domain more intuitively than textual languages, i.e. through a diagrammatic representation in a general-purpose or domain specific notation. Visual languages are used extensively in the domain of process modeling. A summary of widely used visual languages for process modeling is discussed in the context of structured processes in Section 3.2.1. Various other visual languages have been developed to address the need for enhanced flexibility in underspecified, collaborative processes [Swe93, GH98]. Domain-specific visual languages are further used to facilitate the utilization and appropriation of computing technology in scientific and industrial projects. A successful example is National Instruments LabVIEW [Nat08].

Despite the plethora of visual languages addressing different problem domains with different degree of formality, evidences show that even the simplest process modeling requires a perceptible cognitive effort and advanced skills in computer use [MM00]. Furthermore, formal process modeling can distract end users from the actual business tasks as it can be hardly considered as part of their daily activities. Studies on ad-hoc process support consider this limitation and suggest "*the existence of a separate organizational unit for process modeling*" [HMBR05]. Therefore, visual languages are considered here for completeness, as a possible technique to make process models accessible to a larger group of users. However, this group is limited to local developers and programmers. The required knowledge of a given visual notation and the related upfront process modeling make visual languages inappropriate for involving end users in business process composition on a daily basis.

3.1.2.3 Natural Programming

The purpose of natural programming is to build programming environments which better reflect the natural way of thinking of users about the expected program functionality and which facilitate the transformation of the mental plan for this functionality into a system-compatible program description [PM06]. Thereby usability is the primary objective for natural programming environments. The idea of natural programming closely relates to the idea of direct manipulation of data and programming constructs as opposed to formal programming in conventional text-

based programming languages. Natural programming aims at resolving some shortcomings of such programming languages, resulting from their formality and high learning effort, while making programming accessible also for users without programming knowledge. [PM06] for example describes a system which uses an event-based language that features a computational model and provides queries and aggregate operators according to the way that non-programmers expressed program functionality in preliminary studies.

Despite its increased usability, natural programming still requires explicit programming activities by end users, i.e. in a specific programming environment where users can describe the desired program functionality according to the natural programming paradigm. Thus, related work classifies this approach as appropriate for local developers rather than for actual *"non-programmers"*, i.e. end users [SDW08]. Moreover, natural programming requires specific programming environment and does not consider integration in the actual users' working applications. This conflicts with the need to enable unobtrusive process support from the actual end users' working environment that manifested in the empirical studies. Hence, natural programming is not considered as appropriate for end-user driven business process composition.

3.1.2.4 Scripting

Scripting languages enable high-level programming through using existing system components and functionality and combining them to achieve new functionality. In contrast to conventional programming, scripting languages do not use strong typing and do not allow the creation of data structures and algorithms from scratch [Ous98]. Scripting is an embedded capability of many applications. The most popular example is the Microsoft Office suite, where Visual Basic for Applications (VBA) is a universal programming language, allowing experienced users to develop functionality within applications such as Microsoft Word, Excel and Access. In the area of BPM scripting is considered as a generic method to describe business processes in a formal language to a high precision level [Gad08].

Because of their relieved syntax, scripting languages are more-easy to understand than conventional programming languages. However, the level of comprehension is still far from natural language and is inappropriate for technically inexperienced end users (workers). This issue is addressed in [Esl08] through the Social Computation scripting model, which is supported in the ScratchTalk prototype system. The latter enables end users to specify and modify programmatic behavior by using natural language. This is achieved through elaboration of hierarchical plans that represent scripts and consist of goals and sub goals. The runtime model of the system consists of a set of concurrently executing scripts, with asynchronous messaging and synchronized shared variables among them. Although the proposed approach reveals initial steps towards enabling scripting by end users through natural language, it also exemplifies issues regarding the translation of natural language constructs to formal scripts. Such arise e.g. if the same natural language expressions are provided by different users and have different semantics. The basic idea to generate and reuse scripts through techniques that are more-unobtrusive and accessible to end users is applied also in other approaches which are based on PBE [BBCS08].

To sum up, although scripting is considered as a generic method to model business processes in a formal language to a high precision level [Gad08], it is associated with high cognitive effort for studying and applying such a formal language. Considering its complexity level, EUD literature classifies scripting into the expertise domain of local developers rather than in this of *"non-programmers"* [SDW08]. Hence, scripting approaches are considered as inappropriate for end-user driven business process composition.

3.1.2.5 Spreadsheet Programming

Spreadsheets are the most widely spread EUD environment. While spreadsheet applications have gained importance in the business world, the research community is spending increased efforts to

develop methods for keeping spreadsheet development less error prone [BRC06]. Other efforts have been oriented towards supporting collaborative work on complex spreadsheet applications [NM90]. A recent, fuzzy EUD approach related to spreadsheets has been introduced in [SDW08] as the *"accountant paradigm"*. This approach uses the fact that many people understand tabular representation of data without the need for further detailed explanation, and considers that tabular representations can improve the cognitive perception and management of information.

Spreadsheets are a natural EUD environment for a large number of technically inexperienced end users. Therefore spreadsheet applications are particularly interesting from research perspective. However, EUD approaches related to spreadsheets are not considered relevant for end-user driven business process composition due to the restricted focus on managing information in a single application, i.e. spreadsheets, as opposed to the extended scope of a business process which generally involves multiple applications and documents.

3.1.2.6 Component-Based Tailoring

Designing user-tailorable software systems means above all designing flexible systems, which can be adapted or extended in the context of use to match specific user requirements. One way of achieving this is by delivering systems with an extended set of functionality, which is only partially activated and exposed to the end users and which they can activate on-demand. This is the case with customizations which are known e.g. from Microsoft Office applications. However, when it comes to large-scale, enterprise applications, system flexibility is required not only on user interface level, but also on architectural level. For example, system extensions with new modules, incorporating completely new functionality may be required for different business departments and purposes. The software engineering community addresses such issues through component-based models for enterprise applications. In the Java Enterprise Edition (JEE) for example, an enterprise application can integrate a variety of web application, service or library modules. The benefits from component-based architectures are not only in the enhanced adaptability of the system to the changing user requirements, but also in the possibility to distribute expertise to different areas during software development and to facilitate the reuse of software components by reducing the development costs.

The EUD community addresses system adaptation on architectural level through component-based tailoring approaches [WSW06, SQK06], also referred to as component swapping at runtime [SDW08]. These approaches enable users to assemble, re-publish and reuse different application components through visual representations of the components, of the interfaces between them and of the underlying architecture. Through this, tailoring on architectural level can be performed in a controlled manner by the users at run time. A set of functionalities for finding tailoring functions, integrity checking of compositions and exploration of the tailored applications are considered to support component-based tailoring. Nevertheless, the complexity of this approach positions it in the expertise scope of local developers rather than in that of technically inexperienced end users [SDW08]. Moreover, while component-based tailoring promotes software systems' adaptability through deployment and interrelation of different software components, it is the sole functionality of these components that can enable end-user driven business process composition. Therefore, component-based tailoring is considered as a possibility to improve support for business process composition on architectural level. However, such tailoring is not considered relevant for technically inexperienced end users with respect to the composition of business process models.

3.1.2.7 Suitability of EUD Approaches for End-User Driven Business Process Composition

A discussion of available EUD approaches has been provided, looking for concrete ways to enable process composition by end users. Table 3.2 summarizes the discussed approaches, by further referring to the requirements for ensuring a gentle slope of complexity through integrated EUD support within the existing end users' working environments, i.e. email and to-do list

applications (cf. Table 3.1). A further characteristic of EUD approaches in Table 3.2 is their suitability for process composition, which is assessed based on known application domains for the approaches from related literature. Finally, the addressed user types are considered to assess the applicability of the approaches for process composition by actual end users (workers). Approaches that are appropriate for users with less technical expertise are generally considered appropriate also for users with higher IT expertise.

Programming by Example (PBE) supports integration in the end users' working environment, and addresses actual end users. Although research literature does not report application of PBE for business process composition, some similarities of this approach to process mining [vdAW03, vdAW05] can be found. Process mining is discussed further in the dissertation. Therefore PBE is considered suitable for business process composition.

Visual languages do not provide integration support in the actual users' working applications as they require a proprietary environment that is coined to the concrete visual notation and underlying semantics. Although such languages are widely used for process modeling, they are only accessible for users with extensive experience in computer use [MM00] and hence inappropriate for end-user driven process composition.

Natural programming does not provide integration support as it requires proprietary environments that facilitate the programming activity through natural, intuitive program representations. Natural programming is considered appropriate for business process composition as processes can be generally implemented on a low level in any programming language or notation. As natural programming still comes with cognitive effort for learning and explicitly acting in a corresponding programming environment, in the thesis natural programming is considered as inappropriate for technically inexperienced end users. This consideration is supported also in related literature [SDW08].

Scripting functionality is integrated in some widely used software applications such as the Microsoft Office suite. Scripting provides high-level programming capabilities that use and integrate existing functionality of software systems. Scripting is further considered as a possibility to specify business processes at an increased level of details [Gad08]. However, scripting requires knowledge of certain formal notation, which makes it inappropriate for end users. The latter can use PBE as a more unobtrusive technique for script generation, e.g. through recording of macros.

Spreadsheet programming is commonly used by business users in their daily work. However, it restricts to a given application environment, i.e. spreadsheets, and is incapable of capturing input from other applications that are used for ad-hoc work coordination and task management such as email and to-do lists. Hence, spreadsheet programming is not considered appropriate for business process composition.

Component-based tailoring is an architectural approach which relates to system flexibility. It requires an appropriate preliminary system design and is not integrated in conventional users' working applications. Furthermore, this approach is appropriate only for experienced local developers or programmers as it comes with complex implications about consistency of systems' configuration and rollback policies for changes in system components.

Further EUD approaches can be found in related literature, such as *"integrated tailoring interfaces"* and *"accountants' paradigm"* [SDW08]. However, these approaches are still fuzzy and not well defined. The first approach refers to integrating the design-time and runtime view of an application seamlessly, which is largely known as *"What You See Is What You Get"* (WYSIWYG) paradigm in various applications such as spreadsheet applications, word processors or HyperText Markup Language (HTML) editors. The *"accountants' paradigm"* [SDW08] merely refers to the fact that business users understand tabular data without need for further explanation. These approaches are not considered with respect to end-user driven business process composition and not shown in Table 3.2.

To sum up, after a thorough evaluation of available EUD approaches, PBE arises as the most appropriate approach for enabling integrated EUD in the existing users' working environment

towards providing a gentle slope of complexity and overcoming the lacking motivation of end users (workers) to engage with formal software systems. The particular challenge remains to refine the PBE concepts and to develop a methodology for the application of this approach for end-user driven business process composition on enterprise level.

Table 3.2: Suitability of EUD approaches for end-user driven business process composition

Approach	Integration support	Suitability for business process composition	Addressed user type		
			End user (worker)	*Local developer*	*Programmer*
Programming by Example	x	x	x	x	x
Visual Languages		x		x	x
Natural Programming		x		x	x
Scripting	x	x		x	x
Spreadsheet programming			x	x	x
Component-based tailoring				x	x

3.2 User-Centric Process Support

After having evaluated EUD concepts and approaches towards end-user driven business process composition, it is necessary to analyze what approaches are offered in this direction by related research on user-centric process support. Such approaches are discussed in the following.

Composition, management and optimization of business processes are subject to discussion in different research fields. Workflow Management Systems (WfMS) [Hol95] were the first response to the shift from data orientation, which dominated the software industry in the 1970s and 1980s, to process orientation in the 1990s. WfMS provide procedural support for operational business processes [vdAvH02, Wes07].

With the increasing power of information technology over the last decades, new requirements for business process support emerged. This expanded the technological foundation provided by WfMS towards Business Process Management (BPM) systems. While WfMS focus predominantly on three phases of procedural process automation: process design, system configuration and process enactment, BPM systems additionally enable enhanced: process diagnosis, simulation, verification, and validation [vdAHW03]. BPM systems hence extend WfMS and provide more comprehensive support for operational business processes.

An even more extended view on process support than WfMS and BPM systems is considered in Process-Aware Information Systems (PAISs) [DvdAtH05]. A PAIS is a system that has information about an explicit operational process, and both WfMS and BPM systems are considered as PAISs [vdAHW03]. However, PAISs attempt further to consolidate research from different research fields and to consider different types of business processes: (i) person-to-application processes, addressed by WfMSs, (ii) person-to-person processes, which are subject to CSCW research, and (iii) enterprise application integration and business-to-business integration processes. PAIS hence exemplify the broad research scope concerning process composition.

The state of the art analysis presented in the following considers the major challenges for BPM support introduced in Chapter 1. Of particular interest thereby are approaches for composition of different process types – structured, semi-structured and unstructured business processes. These process types are addressed in different research areas ranging from WfMS and BPM systems for structured process support, to CSCW and business-process oriented knowledge

management for supporting unstructured, human-centric business processes (cf. also [Mül05]). Process mining is discussed as a complementary technique for unobtrusive business process composition. The CSCW scope is expanded with related work on evolving workflows, integration of ad-hoc and procedural process support and email-based workflows.

3.2.1 Modeling Structured Processes

Formal systems for automation of operational processes such as WfMS need an explicit process model to operate [vdAvH02, Wes07]. [Gad08] provides a classification of different process modeling methods by differentiating two major types: scripting and visual modeling through process diagrams. As discussed in Section 3.1.2.4, scripting is not appropriate for business process composition by end users due to the associated cognitive effort for learning and applying formal scripting notations. The following discussion focuses on several widely used notations for visual process modeling.

The **Unified Modeling Language** (UML) is one of the visual languages with widest application scope. [EFHT05] exemplifies the use of UML for modeling various aspects of business processes including control flow, process-related objects, and even for modeling the organizational structure of a company and of business partner interactions. [EFHT05] further reveals how UML structure diagrams and interface descriptions can be used to model the process enactment in business integration processes and enterprise application integration processes. Hence, UML as a highly expressive, multi-purpose visual notation which can be used to describe the conceptual setting of an enterprise in terms of organization, processes, resources and software infrastructure. Code generation from UML diagrams can further facilitate the transition from conceptual modeling to implementation of concrete software systems.

Petri nets [PW08] is another popular notation for business process modeling. This notation is particularly useful for modeling systems for which the behavior is dominated by the flow of control or information objects, and which are concurrent, asynchronous, distributed, non-deterministic and/or stochastic. As a visual language Petri nets basically represent directed graphs with two different types of nodes – places and transitions. Petri nets are bipartite, i.e. no arc in a Petri net can connect two places or two transitions. One of the major advantages of Petri nets is that they provide a well-established formalism with mathematical structures and enable formal validation of business process models. Application of Petri nets for process modeling is discussed in [vdAvH02, Des05].

Event-Driven Process Chains (EPC) [KNS92] provide a somewhat more-relieved method for process modeling that is based on the concepts of stochastic networks and Petri nets. In contrast to Petri nets, the EPC notation does not require a strong formal framework as it does not rigidly distinguish between places and transitions. The EPC notation basically consists of different types of nodes and edges. The core nodes are activities, events and connectors. The edges are directed and always connect two elements according to their activation sequence in the business process. All elements can be annotated with textual descriptions to clarify their actual meaning from business perspective. The lack of formality resulting from the relieved syntax and semantics of EPCs can make the formal validation of the process models difficult. This issue is addressed in related work, which describes how EPCs can be mapped onto the more-formal Petri nets [vdA99]. Process modeling with EPCs is discussed in details in [STA05]. The latter study reveals also how different views can be applied during business process modeling where the relationships between different organizational entities, the function flow, the output flow of activities, and the information flow can be modeled independently. These views help to reduce the complexity in process modeling by additionally facilitating the elicitation of requirements for the different areas. The final process model thereby results from the deliberate consolidation of the different views.

Business Process Modeling Notation (BPMN) [OMG06] is a visual notation for process modeling that has emerged from the need for standardization of business process models in order to communicate them better not only internally, i.e. between business users and developers in the

course of workflow projects, but also externally, between different platforms and application vendors. A further incentive for the development of BPMN was to provide a graphical notation for the Business Process Expression Language for Web Services (BPEL4WS) [IMB+03], which can describe it from business perspective. BPMN provides an enhanced graphical notation for modeling of business process diagrams, which consolidates best-practices from various other notations including UML and EPC. While BPMN strives to become the industry standard for business process modeling, it still experiences some difficulties in its expressiveness. Workflow patterns are applied in [WvdAD+06] to assess the suitability of BPMN for process modeling. The latter study reports difficulties for the usage of BPMN for process modeling resulting from the lack of *"commonly agreed-upon formal semantics"* and *"execution environment"* for BPMN. These findings are confirmed in [RIG07] where additional issues are raised with respect to the usefulness and expressiveness of BPMN from a user's and from a developer's perspective.

The notations discussed above are only a small excerpt from the plethora of general-purpose and domain-specific visual languages for process modeling. An extended discussion on modeling languages can be found in [Gad08, Jor04]. Such is not provided here, as all languages for explicit modeling share the same intrinsic barrier regarding end-user driven process composition, i.e. these languages require knowledge of a concrete visual notation and the associated semantics. Empirical findings from research on process tailoring through visual notations [MM00] show that such tailoring requires advanced skills in computer use. As technically non-skilled business users lack such skills, the above visual modeling approaches are inappropriate for end-user driven process composition. The need for unobtrusive approaches arises, that can address business users without confronting them with upfront process modeling notations and environments.

3.2.2 Process Mining

Process mining focuses on the extraction of information about the actual execution of processes from event logs [vdAW03, vdAW05]. The purpose thereby is to facilitate process analysis, monitoring and improvement. Process mining does not rely on the existence of a WfMS. It can be applied to various types of transactional systems such as WfMSs, Enterprise Resource Planning (ERP) systems, Customer Relationship Management systems, Supply Chain Management systems, and even on groupware systems for mining ad-hoc processes from event logs on ad-hoc collaboration [DHvdA05]. Process mining can be further used for uncovering organizational relationships and mining of social networks [vdAS04]. This approach further enables the derivation and iterative refinement of explicit workflow models from implicitly captured activities in formal systems [Kle04, HHK04].

Process mining is capable of providing unobtrusive support for business process composition and tight integration in the actual working environment of end users. With this respect it has some similarity to PBE, as both approaches generate software artifacts by capturing actual user behavior. However, process mining has some intrinsic disadvantages from EUD perspective. When discussing meta-design and the SER process model (cf. Section 3.1.1.3) [FGY+04] clearly emphasizes on the importance to *"provide the socio-technical environment for stakeholders to become informed participants"* in tailoring of software artifacts. However, process mining approaches produce process models from user activities by not keeping the users informed of the emergent processes. Therefore, users are not able to establish a relationship between an emerging formal model and the actions which they perform. In that sense end users do not tailor the processes proactively but can rather view the final result from the activities, which they have performed, in a formal notation, which they may not understand. Hence, process mining does not provide a gentle slope of complexity, because it does not consider gradual involvement of end users in process tailoring. As a result, process mining can be rather considered as a technique for process analysts and designers and not for end users. In contrast to that, PBE is performed by end users in an informed manner, i.e. with the expectation of producing a useful software artifact which can be further adapted and reused in recurring cases. Such is for example the generation

and reuse of macros and the adaptation of macros through editing their code.

Furthermore, process mining is strongly system-dependent and inherits the constraints of the underlying formal systems. These systems may not capture the complete information flow in enterprises. Evidences show, that a lot of agreements and work coordination are performed in informal communication environments such as email [BDHS03, BDH+05]. Studies, focusing on mining of ad-hoc processes emphasize on the need to support unstructured and semi-structured business operations [DHvdA05]. However, the approach proposed in the latter study is coined to the concrete system environment and to the underlying data model of a process-enhanced groupware application. This approach does not address the extensive usage of email for supporting informal work (R2), which is evident in related literature [BDHS03, BDH+05] and which has manifested also in the empirical studies (cf. Chapter 2). The need for user-centric approaches arises, which can enable *"informed participation"* of end users in business process composition by fostering *"social creativity"* [FGY+04] and allowing domain experts to proactively drive process optimization in enterprises from their common working environment.

3.2.3 Composition of Semi-Structured and Unstructured Processes

In [DvdAtH05] processes are classified with respect to their structure and predictability as *"unframed, ad hoc framed, loosely framed and tightly framed processes"*. The latter study considers as *unframed* such processes that cannot be associated with an explicit process model, like e.g. collaborative processes which are supported through groupware applications. Thereby [DvdAtH05] further suggests that *"unframed processes can lead to framed ones, and there is no clear-cut boundary between these categories"*. Conventional WfMS and BPM systems focus on structured (*tightly framed*) processes and require explicit models to operate [vdAvH02, vdAHW03]. On the other hand, support for knowledge-intensive, unstructured (*unframed* or *ad-hoc framed*) processes is addressed from various research areas such as CSCW and knowledge management. A discussion of the suitability of these approaches for end-user driven business process composition is provided in the following by considering the identified fundamental EUD concepts and the requirements from the empirical studies (cf. Table 3.1).

3.2.3.1 Knowledge Management Approaches

The reuse of emerging task hierarchies within a global enterprise infrastructure is often described as one of the major possibilities to support underspecified, human-centric processes. Intrinsic challenges for the next generation task management and BPM are discussed in [RRMvdA05], where the generation, recognition and application of reusable *"task patterns"* and *"process patterns"* is suggested as an alternative to static workflows. The task pattern technique is considered in further studies [GOR+07] describing basic directions for the utilization of task-based approaches to support users in intensive, unstructured knowledge work. Although from theoretical perspective the proposed patterns' approach is feasible for ensuring user-centric process support, approaches for engaging end users in the composition of business process models through the generation, adaptation and reuse of task patterns are still missing.

A similar approach for user-centric process support through guidance and reuse of process knowledge is proposed in [HRD+06]. This approach enables users to manage ad-hoc tasks in hierarchical to-do lists, where further document-based and task-based proactive information delivery for recurring cases and instance-based task reuse are enabled. The proposed approach incorporates further advanced techniques for building personal knowledge spaces and wiki-based collaborative document spaces. This approach corresponds to a comprehensive strategy for business process oriented knowledge management [HMBR05]. The latter study raises the issue that explicit process modeling can result in overhead for end users as it can be hardly considered as part of their daily activities. Therefore the study suggests *"the existence of a separate organizational unit for process modeling"*, yet confirming the disruption between business users

and business technology experts, i.e. process designers and developers. This amplifies the importance of ensuring a gentle slope of complexity [MCLM90] in user-centric process composition environments and emphasizes on the lack of support for seamless transition from informal coordination, ad-hoc decision support and flexibility in knowledge-intensive processes to formal process modeling and automation of structured processes.

3.2.3.2 Specificity Adjustment - From Ad-hoc to Procedural Process Support

A comprehensive approach, addressing the gap between completely ad-hoc and procedural process support is discussed in [Ber00]. This approach enables users to create shared, distributed-accessible, hierarchical to-do lists, where different process participants can access and enrich task resources and information. This approach enables *"contextual basis for situated improvisation"* by enabling delivery of *"process models, process fragments, and past cases"* similarly to the previously introduced task-based approaches for ad-hoc process support from the knowledge management domain [RRMvdA05, HRD+06]. However, the approach introduced in [Ber00] further enables process definition in different specificity spectra and transitions between these spectra during process execution. The specificity spectra proposed in [Ber00] are: (i) *"provision of context"*, where users specify hierarchical to-do list and can attach files (as resources) to each of the to-do items (tasks) in these lists; (ii) *"monitoring of constraints"*, where users can specify constraints for to-do items such as due dates; (iii) *"planning of options"*, where users can set options for the execution of the tasks based on the provided constraints in order to receive notifications and suggestions by the system about possible constraint violations; (iv) *"imperative scripts/directions"*, which use specified constraints by directing the task execution and associated resource usage automatically as opposed to the suggestive behavior of the options planning. Although this approach supports advanced flexibility in process specification and execution, it has some disadvantages with respect to end-user driven business process composition based on task management. Concretely, it cannot be assumed that business users will be active in all specificity spectra during their daily work per se. Related literature on personal task management reveals that users often record to-do's in an underspecified manner [BDG+04]. Hence, the provision of context may remain the only specificity spectrum in which end users act. While constraint definition can impose structure in the evolving tasks it cannot by relied upon for engaging end users in composition of structured workflows. The proposed specificity spectrum of automated *"imperative scripts/directions"* also remains inaccessible for technically inexperienced end users because of the required technical knowledge and explicit modeling effort to define such scripts. Hence, approaches are needed, which can enable transition from ad-hoc to structured processes by using only data from the context provision spectrum.

3.2.3.3 Evolving Workflows by User-Driven Coordination

A further approach that focuses on user-centric support for underspecified, human-centric business processes is presented in [Her00]. The latter study introduces the concept of *"evolving workflows"* which have the purpose to ensure that *"those employees who execute the task are the same ones who develop fragments of workflows, refine them and eventually combine them to create a network of coordination"*. Evolving workflows are motivated among others through the possibility to make workflow projects less error prone by involving end users directly in the composition of business processes and reducing the risks of misinterpretation of user requirements. For realizing evolving workflows [Her00] suggests immediate, embedded support during the usage of operative software applications from the common users' working environment. This approach hence addresses implicit process composition from the context provision specificity spectrum as opposed to explicit specification of constraints and process formalization discussed in [Ber00]. As a possible realization [Her00] proposes the use of add-on components, which register users' activities, used documents, involved stakeholders and other

context information. [Her00] further proposes that based on the collected information, the system should be able to deliver rough process diagrams, which can be subsequently adapted and extended by users. While the described guidelines for technical realization resemble process mining [vdAW03, vdAW05], no concrete approaches for the realization of evolving workflows have been identified so far. Nevertheless, the concepts behind evolving workflows find strong support in the EUD domain, e.g. with respect to rendering of the process tailoring to the end users, who are also the owners of problems, and enabling iterative refinement of process models as underspecified software artifacts [FGY+04]. The concept of evolving workflows hence amplifies the need for approaches for end-user driven business process composition through unobtrusive support in the users' working applications.

3.2.3.4 Interactive Process Models

The idea of iterative refinement of underspecified process models from evolving workflows is developed in a further approach for user-centric support of knowledge-intensive, creative processes [Jor04]. The latter study proposes the use of interactive process models, which are incomplete, partially ordered and partially decomposed and integrate modeling (articulation) and enactment (activation) of unique process instances while the process evolves. However, unlike the concept of evolving workflows that emerge through collecting system data from underlying end users' working applications [Her00], process support through interactive process models requires initial process models in a simplified visual notation, which can be manually interpreted and adapted by end users during process execution. Although interactive process models are intended for actual end users (managers and workers), [Jor04] reports that during the evaluation studies *"Feedback from users indicates that for some people, especially in the initial phases, process models are still too complicated"*. These findings agree with related literature showing that visual process modeling requires extensive technical skills [MM00]. On the other hand, the increased acceptance towards the management of individual work items in task lists without any visual notation that is reported in [Jor04] confirms the unobtrusiveness of direct manipulation techniques as these can be found also in related EUD literature [PM06]. Because of the required explicit, visual process models the discussed interactive process models approach [Jor04] does not offer adequate support for end-user driven business process composition with respect to ensuring a gentle slope of complexity [MCLM90]. Approaches are needed, which enable process composition through direct manipulation of process representations in the common end users' working environment.

3.2.3.5 Case Handling

A further approach to support flexible and knowledge-intensive business processes is through case handling [vdAWG05, Ber05]. Case handling does not impose strict prescriptions of how a user should act in an ad-hoc process, but provides assistance by showing how a business goal can be achieved so that the user can select the appropriate course of action that best suites them. Thereby activities are not considered as atomic like in WfMSs, but rather as chunks of work, incorporating certain interruption points where work can be transferred between different stakeholders, e.g. an activity of filling out a (web) form can be interrupted after filling only part of the fields and then transferring to another person. From case handling perspective a process can be seen as a recipe for handling cases of a given type. The cases refer to a specific business goal and can follow certain patterns. Case handling considers that exceptions are a rule in ad-hoc processes and therefore proposes a minimization of the precedence relations among activities similarly to evolving workflows (cf. [Her00]). Instead of strict precedence relations, case handling environments usage roles for rerouting of activities and for exception handling. As the control flow is unpredictable, data objects are used instead of control objects (e.g. tasks) to determine the state of cases.

Despite the increased flexibility, case handling systems share some of the disadvantages of conventional workflow management approaches with respect to end-user driven business process composition. Namely, case handling systems require explicit role modeling for each process object [vdAWG05, Ber05]. This modeling is basically performed by a process designer who has knowledge of the case handling paradigm. The required technical skills for setting up case handling environments makes these inappropriate for process composition by end users.

3.2.3.6 Email-Based Workflows

Evidences from related literature show that user strategies for organizing daily activities are far from any process or case definition context and mostly rely on common office tools such as email [GRM+04, BDH+05, SIT06] or personal to-do lists [BDG+04]. Email has become the natural habitat of knowledge workers and the primary environment for Personal Information Management (PIM) [DB01]. Personal task management through (hierarchical) to-do lists complements email usage by explicating work items, structuring work, and facilitating management of time critical activities, e.g. through priority and status monitoring. To-do lists are seen as a natural environment for managing unstructured, knowledge-intensive processes which is accessible and intuitive for end users without technical skills [Ber01, Jor04, HMBR05, HRD+06]. The extensive usage of to-do lists and email for management of personal and group activities manifested also in the preliminary empirical work (cf. Chapter 2). Hence, support of end-user driven business process composition through a gentle slope of complexity can begin on personal level, i.e. with supportive activities for PIM in common users' working environments such as to-do lists and email (cf. Table 3.1).

Related literature reports approaches that facilitate PIM with respect to email [BDHS03] and task management in to-do lists [BDG+04]. However, these studies focus on increasing the individual performance and do not cross the boundaries of the personal workspace towards composing end-to-end enterprise processes. Further approaches enable management of ad-hoc tasks in personal as well as organizational settings [Cha08, HRD+06]. However, the latter approaches rather focus on supporting individual ad-hoc tasks in a group context rather than on involving end users in composition and adaptation of explicit business process models.

Email-based workflows cross the boundaries of the personal workspace and integrate to-do lists and email for supporting agile business processes [ADMG97]. The latter study presents a system, which is based on computational email [Bor92] and uses three different kinds of active software objects that are embedded in email messages: (i) *"mail-robots"*, which are specialized portions of code that are executed without directly involving the user, either at email delivery time or at receipt time; (ii) *"agents"*, which are multipurpose and event-driven portions of code devoted to handling the objects of the email-based workflow system and their relationships; (iii) *"applets"*, which are based on Java technology and enable a broader set of functionalities. Although the email-based workflow approach proposed in [ADMG97] seems highly promising for enabling business process composition by end users from common, light-weight working applications, some intrinsic deficiencies arise for this approach from EUD perspective. On the first place, [ADMG97] does not discuss who is responsible for specifying the used active software objects. Coding of *mail-robots* or *agents* can be hardly performed by end users even through scripting techniques (cf. Section 3.1.2.4) not to speak of the more-complex *applets*. Hence, the presented system needs to be configured for usage by technically skilled local developers or programmers. Second, no mechanisms for decoupling emergent email-based workflows from the system as explicit process models are discussed, or how such models can be exchanged, adapted and reused. As end users have different level of technical expertise and attitudes towards maintaining process data it is important to consider possibilities for SER [FGY+04] of user-defined task structures for their iterative refinement (cf. Table 3.1). Hence, approaches for end-user driven business process composition are needed, which leverage the existing end users' experiences with to-do lists and email, and which use existing software

artifacts from the users' working applications such as task items and email messages to compose business process models.

3.2.4 Suitability of User-Centric Approaches for Process Composition by End Users

An extensive discussion of available approaches for composition of structured, semi-structured and unstructured processes has been provided in the previous sections. The discussed approaches are summarized in Table 3.3 with respect to the identified fundamental EUD concepts and the requirements from the empirical studies (cf. Table 3.1). The suitability of these approaches for end-user driven business process composition is summarized in the following.

Table 3.3: Assessment of approaches for user-centric process support

Approach	**Addressed process types**			**Supported EUD concepts & related requirements**						
	Unstructured	Semi-structured	Structured	Gentle slope		Reduced expertise tension	SER			Process tailoring as collaboration
				R1	R2	R3	R4	R5	R6	R7
Visual process modeling		x	x			x				
Process mining		x	x			x				
Knowledge management approaches	x	x		x	x	x	x			
Specificity adjustments	x	x	x	x	x	x	x			
Evolving workflows	x	x		x	x					
Interactive process models		x	x			x	x			x
Case handling		x				x	x			
Email-based workflows	x	x		x	x	x				

Visual process modeling approaches have been introduced with respect to structured, predefined processes that can be automated through WfMSs or BPM systems. Further visual approaches from the EUD domain enable modeling of semi-structured, collaborative processes [Swe93, GH98]. Visual process models generally support overview and guidance during run-time (R3), depending on the software system that interprets them, e.g. a WfMS. However, with respect to the process design phase, explicit visual process modeling requires knowledge of a given modeling notation and is associated with perceptible cognitive effort. The required advanced skills in computer use [MM00], conflict with the need to provide a gentle slope of complexity for process composition by end users. Further, although SER is supported in some visual modeling environments through collaboration support for editing and refinement of process models by developers [GH98], SER approaches for process model refinement by end users are lacking. Visual modeling environments do not enable process tailoring as collaboration either, as technically inexperienced end users are incapable of understanding and dealing with formal modeling notations per se [MM00].

Process mining approaches can be applied for the composition of semi-structured and structured processes from logged data on events in collaboration tools or enterprise systems. However, process mining involves end users in process composition in a rather passive manner,

by not keeping them aware of the emerging processes and not fostering their proactive attitude towards tailoring the emerging process definitions. The systems, from which processes are captured, do not support modeling and activation of process models per se. The captured process definitions are mostly provided in a formal (visual) notation [Kle04, HHK04] for analysis by technically skilled persons. Thus, process mining separates process execution from process design from end user's perspective and does not provide a gentle slope of complexity for involving end users in process composition. Process mining techniques generally enable transparency (R3) as process models can be derived according to the available logged data in running processes to monitor these processes. However, for user-centric monitoring support, it has to be additionally ensured that the derived process models can be understood by end users. Some SER capabilities are provided by process mining approaches for workflow designers that allow them to evaluate deviations from predefined workflow models and to optimize these models [Kle04]. However, reuse and adaptation of best-practices (R4, R5) are not enabled for end users. Furthermore, process tailoring as collaboration is not supported, as the end users actually do not purposefully compose a process, i.e. there is no cognition about the influence of users' actions on the emerging process definitions. This opens space for misinterpretation of "mined", user-defined process models by process designers and developers.

Knowledge management approaches for supporting completely ad-hoc and semi-structured processes have been discussed [RRMvdA05, GOR+07, HMBR05, HRD+06]. These aim at the optimization of the individual activities in informal, knowledge-intensive processes and have a strong user focus. These approaches propose extensive use of personal task management and collaborative workspaces [HMBR05] and integration of the personal task management perspective into an enterprise BPM perspective [RRMvdA05]. Although the latter approach considers email as a competitive environment for BPM-centric task management support, later work acknowledges the need to integrate email (R2) as a substantial part of the users' work practice [GOR+07]. All approaches consider the usage of (hierarchical) to-do lists (R1) for ad-hoc task management. Hence, knowledge management approaches can provide a gentle slope of complexity by not confronting end users with explicit formal process modeling notations. The approaches further ensure increased transparency (R3) and reuse of process knowledge (R4). Yet these approaches lack support for SER with respect to (R5) and (R6) as they do not discuss possibilities for structured comparison between evolving processes and best-practice definitions or between different best-practice definitions [RRMvdA05]. Process tailoring as collaboration is also not considered (R7) as all discussed knowledge management approaches dissociate from formal, rigid process modeling due to the inability of structured process models to support knowledge-intensive processes.

Specificity adjustment in different spectra towards composition of unstructured, semi-structured and structured processes is proposed in [Ber00]. This approach is based on hierarchical to-do lists (R1) and email (R2), and is capable of providing a gentle slope of complexity for process composition by end users. Transparency (R3) and reuse of process knowledge (R4) are also considered by the latter approach. However, this approach does not consider possibilities for a structured comparison between reused process templates and running processes (R5) or between different versions of a process template (R6). Furthermore, this approach does not address process tailoring as collaboration (R7) as for process automation if uses imperative scripts that are based on constraints in ad-hoc task hierarchies. The definition of such scripts is inappropriate for end users due to the required increased cognitive effort and technical skills. A transformation of user-defined to structured process models that can be extended with complex functionalities e.g. for event handling or transaction triggering by developers is neither discussed in [Ber00].

Evolving workflows is an approach that addresses support for unstructured and semi-structured processes [Her00]. This approach indicates the need to ensure a gentle slope of complexity for end-user driven business process composition, i.e. through embedded support in existing software systems (eventually R1 and R2). The approach also suggests that user-defined

process models should be extensible by developers and process designers (R7). However, the approach of evolving workflows is still fuzzy and incomplete. No concrete methods, models or tools for its realization are available, which can help to estimate its correspondence with the fundamental EUD concepts and related requirements.

Interactive process models are capable of supporting semi-structured and structured processes [Jor04]. They enable transparency (R3) in evolving processes and reuse of previous process knowledge (R4). However, interactive process models require preliminary modeling in a simplified modeling notation. As a result they do not fully address the gentle slope of complexity for process composition by end users. Analytical support for SER through structured comparison between reused best-practice templates and running processes (R5) and between different best-practice variations (R6) is also not considered by this approach. Interactive process models demonstrate some potential for supporting process tailoring as collaboration (R7) as they comply with a certain formal notation, which can be used by process designers and developers for complex modeling activities. However, [Jor04] reports that this notation is difficult to comprehend by end users without technical skills.

Case handling supports semi-structured processes [vdAWG05, Ber05]. It has similar deficiencies with respect to end-user driven business process composition as interactive process models and conventional workflow management approaches. Namely, case handling requires preliminary, explicit process models in terms of case definitions, which are based on generic roles and require cognitive modeling effort. Hence, case handling does not provide gentle slope of complexity for process definition by end users. Through the provided guidance and reuse of previous experience case handling approaches are capable of addressing (R3) and (R4). However, no SER support is considered with respect to tracing differences between running processes and reused case definitions (R5) or tracing the evolution of case definitions (R6). Case handling further does not support process tailoring as collaboration (R7) is it does not consider a transition from user-defined, weakly-structured process models to formal process models for automation of rigidly recurring processes.

Email-based workflows provide support for unstructured and semi-structured processes from common end users' working applications such as to-do lists and email [ADMG97]. Thus, email-based workflows enable a gentle slope of complexity with respect to (R1) and (R2). Transparency (R3) is also provided in email-based workflows. However, extraction, adaptation and reuse of previous process knowledge (R4) are not addressed in [ADMG97]. Therefore also no advanced SER capabilities (R5, R6) are enabled in general. Further, email-based workflows do not consider process tailoring as collaboration (R7) as they do not provide export of user-defined process models to formal process modeling notations towards model extensions by developers and automation of rigidly recurring processes.

To sum up, the discussed approaches for user-centric process support are unable to support end-user driven business process composition in all required aspects. This exemplifies the research gap and the need for novel approaches for process tailoring by end users. The state of the art analysis has provided strong support for the use of hierarchical to-do lists (R1) [ADMG97, Ber00, BDG+04, Jor04, RRMvdA05, HRD+06] and email (R2) [ADMG97, GRM+04, BDH+05, SIT06] for involving end users in process composition by providing a gentle slope of complexity. Generic concepts for facilitating exchange, adaptation and reuse of process knowledge (R4) have also been identified in different threads of research related to user-centric process support [Ber00, Jor04, RRMvdA05, HRD+06]. In the dissertation the notion of *task patterns* introduced in [RRMvdA05, GOR+07] is adopted, as the task patterns approach clearly addresses the need to reconcile the personal task management perspective with the enterprise BPM perspective towards end-user driven BPM. In the dissertation task patterns are considered as an approach to enable SER of user-defined process models, whereas a particular challenge remains to develop concepts for the definition, adaptation and reuse of task patterns by end users by considering relationships between running processes and originating task patterns (R5) and between different task pattern

variations (R6). The definition of task pattern used in the thesis is provided in Chapter 4. A further challenge remains to enable process tailoring as collaboration between end users and technically skilled process designers and developers. With this respect a transition from user-defined to formal process definitions is necessary, which is not considered in related literature.

3.3 Summary

This chapter has presented the state of the art analysis related to end-user driven business process composition. Two major perspectives have been considered. First, related EUD literature has been discussed to identify fundamental concepts related to tailoring of software artifacts by end users. These concepts have been translated for the domain of business process composition. Appropriate meta-approaches for end-user driven business process composition have been identified and combined with the requirements from the empirical studies discussed in Chapter 2.

An assessment of concrete EUD approaches for the creation and modification of software artifacts followed to identify possible candidates for enabling end-user driven business process composition. The assessment has leveraged Programming by Example (PBE) as the most suitable EUD approach, which is able to provide integrated support for process composition in the actual end users' working environment by decreasing users' motivational barriers for engaging in process tailoring.

Further, approaches for user-centric process support from different research areas have been discussed with respect to their suitability for end-user driven business process composition. The discussion has been underpinned by the identified fundamental EUD concepts and the associated requirements from the empirical work. Composition of structured processes has been discussed with respect to known approaches from WfMS, BPM systems and PAIS. Support for semi-structured and unstructured processes has been discussed based on related literature from different domains such as CSCW and knowledge management. Benefits and deficiencies of known approaches have been identified to motivate the need for novel frameworks for end-user driven business process composition.

CHAPTER 4: Task Management Model

This chapter presents a task management model, which is the first major scientific contribution of the thesis. The task management model provides a formal specification of a set of concepts and relations that enable aggregation and handling of data for end-user driven business process composition based on personal task management both, in personal as well as in organizational settings. In order to clarify the developed task management model, this chapter first provides an overview of the generic approach for end-user driven business process composition. The approach is detailed through the holistic concept presented in Chapter 7. A task management model is required in order to define how process instances emerge through personal task management activities, i.e. what are the underlying entities for composing a business process in terms of control flow, document flow, event flow and human actors' information.

The thesis discusses two major aspects of the task management model: (i) *runtime task management model*, which defines task management at instance level, i.e. this model defines the concepts and relationships for supporting emerging process instances that attribute to a concrete execution context, and (ii) a *task pattern model*, which defines task management at schema level, i.e. defines task patterns as reusable task models for composing weakly-structured processes. The presented task management model is motivated through related work and considers the fundamental EUD concepts and requirements from the previous chapters.

4.1 Task Models and Business Process Models

The term "*task*" has partially overlapping, but different meanings in the areas of Human-Computer Interaction (HCI) and Business Process Management (BPM). HCI research considers tasks as low-level interactive activities, such as providing generic system input (e.g. writing text in word processors or web forms), browsing or searching (e.g. in text documents or on the web), and aims to facilitate the understanding of how interactive user tasks can be supported in an efficient and user-friendly manner. Different methods for task modeling and analysis have been developed for this purpose. A widely spread generic method for investigations of human performance that involves computer systems is Hierarchical Task Analysis (HTA) [AS00, Ann04]. HTA focuses on the goals that should be achieved through a given task and enables goal decomposition and definition of operations and sub-operations for achievement of the identified goals. Plans are further considered, which define rules for the operations' sequence in different circumstances. The principles of HTA about goal definition and decomposition have been adopted by further HCI methods for task analysis such as Goals, Operators, Methods, and Selection rules (GOMS) [JK06a, JK06b, Joh03]. These methods originate from applied information-processing psychology [CMN83] and provide a cognitive model of user behavior which focuses on the goals that users want to achieve through interacting in computer systems. Limitations of GOMS like strictly sequential task ordering, and weak tool support for pragmatic application in real-life settings are addressed in a further task modeling framework called Concur Task Trees (CTTs) [PMM97, Pat04]. CTT enables system designers to describe the logical activities that an interactive application should support and facilitate model-driven software engineering from requirements analysis [MPS02] to user interface design [KK05]. The CTT notation is highly expressive and includes temporal relationships between tasks, hierarchical work break-down, and different task types such as interactive, human or (automated) application tasks. Further approaches like Groupware Task Analysis (GTA) [vdVLB96] combine task analysis methods from HCI with ethnographic methods as used in Computer Supported Cooperative Work (CSCW) towards comprehensive methodologies for requirements analysis and design of

groupware applications. GTA has been adopted and extended by further methods for design of complex interactive systems [vdVvW04]. A detailed discussion and comparison of different task modeling approaches for user interface design is provided in [LV04].

Some similarities can be found between task modeling approaches from the HCI area and business process modeling approaches from the area of BPM. For example, the task types of CTT: user task, interaction task, and application task [PMM97], share similar meaning correspondingly with the basic BPMN task types – user task, manual task, and service task [OMG06]. BPMN gateways on the other hand resemble task flow gateways from the task modeling methods adopted by GTA [vdVLB96]. However, in contrast to the area of HCI where task models refer to low-level interactive activities and are used for designing interactive systems, BPM literature considers tasks as high-level steps in business processes [vdABV+99, vdAvH02]. A business process is generally considered to handle a specific case, e.g. a tax declaration or an insurance claim. Thereby human activities comprise cases which are handled through performing sequences of tasks by using appropriate resources [vdABV+99]. The task sequences determine the overall process flow, whereas business process models do not provide detailed specification of the necessary interactions for performing a given task on system level. Business process models preserve a certain level of abstraction to be able to support process automation on different technological platforms. The models need to be interpreted by a BPM system or by a workflow management system, which links human participants with the appropriate applications and supplies them with the right information to perform their tasks [vdAvH02].

To sum up, task modeling approaches from the area of HCI do not share the purposes of business process modeling and do not provide adequate techniques to represent process aspects such as control flow, data flow, event flow and involved human actors. Merging both modeling approaches - task modeling from the HCI field, and business process modeling, can lead to new methodologies for designing customizable BPM systems. Such systems can benefit from model-driven user interface design and provide BPM environments with adaptable, process-specific and task-specific interfaces. Such considerations exceed the scope of the current thesis.

The thesis is focused on the creation and adaptation of business process models by end users and adopts the notion of *"task"* introduced in BPM literature [vdABV+99, vdAvH02] (cf. Definition 1.2). A task is considered as a fine-granular building block of a business process, which defines a generic business activity that should be performed to achieve the desired business goal of the process. Thereby no interactions on system level for executing a given task are specified. For example, typical tasks in a process for selling goods to a customer on credit would be to check the customer's credit history and to get approval from management for credits above a given limit. The corresponding task models in the business process model can define parameters or business rules for these tasks but may not define interactions on system level, such as e.g. input of customer name or credit amount in interactive forms, to keep the process model generic. The interactions are implementation-specific and depend on the concrete system on which these tasks are executed.

4.2 Task Patterns as Knowledge Artifacts for Ad-Hoc Process Support

Conventional workflow management systems are bound to rigid process definitions and do not provide sufficient flexibility to support ad-hoc, knowledge-intensive processes [Sch01]. For supporting such processes [Sch01, Sch03] propose a task-based approach, where processes emerge as hierarchical task structures that are composed during process execution from ad-hoc, user defined task representations. Available task models for recurring cases are provided as building blocks for emergent process models at runtime, whereas task instances represent copies of the reused task models within a concrete process execution [Sch03].

The idea of dynamically composing ad-hoc processes by using reusable task structures is further adopted in [RRMvdA05]. The latter study proposes the use of *"task patterns"* and *"process patterns* to support ad-hoc processes by facilitating the reuse of process knowledge. [RRMvdA05] proposes separation of work knowledge into independent *"Task Information Units (TIUs)"* which *"can describe data aspects, e.g., concrete customer data, but also process aspects, e.g., steps required to file a patent"*. [RRMvdA05] further suggest that users deal with task patterns and task related information in that they provide information to characterize the task they want to accomplish, and based on this information the system provides *"different kinds of TIUs: (1) Process Patterns that can be used to structure the task into suitable sub-tasks; (2) Task Related Information, which support the execution of task, e.g., regarding experts who can be consulted or external services on which the user can draw; and finally (3) relations between these information units and the task or specific sub-tasks of a chosen process pattern"*. To decrease the effort that end users need to spend for organizing the work and for providing task descriptions, [RRMvdA05] further consider observing users' desktop activities, interactions with applications such as e.g. email, browser, text editors or document repositories to build and leverage a user's context and try to figure out a generic task, i.e. *task pattern* that the user is executing. Ad-hoc processes are then supported through capturing and providing task patterns for reuse in recurring business cases. For managing task patterns [RRMvdA05] further suggest that *"task patterns require repositories containing descriptions of cases, which have been executed, including all relevant task constituents. Context, goal, and planning information must be stored and can be used to identify appropriate task patterns"*. [RRMvdA05] uses the terms *"TUIs"*, *"task patterns"* and *"process patterns"* in an interleaved manner without providing explicit, strict definitions of a process, a task, a process pattern or a task pattern. Both, tasks as well as processes can be fully executed in a common application environment, and each process as well as each task has a given goal. However, [RRMvdA05] does not discuss any boundaries where a task with a generic goal becomes a process, e.g. through further decomposition and delegation of sub-tasks, or where a task pattern becomes a process pattern, i.e. through abstraction and generalization of the corresponding goal and contained information. On the one hand, the term "task pattern" seems to relate to reusable information units that are based on captured desktop activities of a system user and provide information about used applications and executed interactive tasks, such as opening a document or browsing on the internet. On the other hand, task patterns seem to relate to generic task structures that define the overall goal that has been pursued in a given business case, and the sub-steps that have been performed to achieve this goal. Thus, task patterns can serve also as process patterns or task models [Sch01, Sch03], i.e. as building blocks that are used to compose dynamically ad-hoc processes during their execution based on previous experience.

The concept of task patterns for supporting ad-hoc, knowledge-intensive processes introduced in [RRMvdA05] is further developed in the NEPOMUK project [NEP06]. [GOR+07] proposes dynamic composition of ad-hoc processes in the form of task hierarchies. This composition is based on a task management model where *"Task Patterns describe a kind of active task templates that provide information that helps users to organize their own task. A task pattern can be regarded as an abstraction of a class of similar cases and thus describes a kind of best practice for the execution of specific tasks. In this respect, a task pattern can contain all kind of reusable information resulting from cases"* [GOR+07]. The latter study considers various static task pattern information such as possible sub-tasks, dependencies between sub-tasks, decisions, and completion measures, and dynamic task pattern information such as information objects and statistics. [GOR+07] further suggests, that *"task patterns are created by an abstraction process and task instances are created from patterns in an instantiation process"*. Thus, task patterns serve as task models (cf. also [Sch03]) that can be used to compose task hierarchies, i.e. ad-hoc processes, by reusing previous process knowledge such as task decomposition, associated information artifacts and expertise.

Further developments of the concept of task patterns for supporting ad-hoc, knowledge work

can be found in [RCKM06, OGR07, JGOR07, Sch09]. The latter studies leverage the role of task patterns as reusable, explicit task structures that capture personal and organizational knowledge on recurring business cases in the context of pattern-based task management. [Sch09] describes an enhanced personal task management system that has been developed in the NEPOMUK project [NEP06] to provide support for task patterns as reusable knowledge artifacts in ad-hoc processes. In the described system, task patterns are developed explicitly based on task representations in a hierarchical to-do list [Sch09]. No automated detection of interactive tasks from users' desktop activities is performed for recognition and capturing of a task pattern as initially proposed in [RRMvdA05]. Thus the role of task patterns is primarily to serve as reusable building blocks for ad-hoc processes that are based on explicit task representations in a task management system and provide personal and organizational process knowledge.

The thesis focuses on business process composition based on personal task management. Similarly to [RCKM06, OGR07, JGOR07, GOR+07, Sch09], a task pattern is considered as reusable task structure that describes how a given generic business task can be accomplished. The concrete task pattern definition that is adopted in the thesis is provided later on. Here it is important to stress that task patterns serve as building blocks for weakly-structured, knowledge-intensive business processes. These processes are based on hierarchical task decomposition rather than on formal workflow models [Sch03, RCKM06] as discussed further in the thesis. For example a task pattern for selling goods on credit can contain sub-tasks for checking the customer's credit history and acquiring approval from a senior manager for credits exceeding a given limit, as well as information about required documents and related experts. This pattern can be used as best-practice in an ad-hoc sales process, if the customer is not able to pay the complete amount of the ordered goods. The task pattern can be combined with the task structure and information from the conventional sales process to compose a sales process involving ordinary payment and credit. Thus the adopted notion of task patterns refers to task patterns as task models, i.e. as building blocks for dynamic composition of ad-hoc business processes.

The notion of task patterns that is adopted in the thesis has to be clearly distinguished from task patterns for interactive systems design. Such patterns are discussed in HCI literature as reusable structures for task models of interactive applications [Pat00, GSSF04, PB04, Sin04] and describe recurring situations during the user interaction with a software system. For example an *"Evaluation Task Pattern"* is used to model situations in which the user selects a set of data that needs to be evaluated and inserts parameters required during the evaluations [Pat00]. Other typical examples for such task patterns are search and login operations or filling out a form [GSSF04]. The notion of task pattern in the area of interactive systems' design thus refers to recurring user activities at interaction level rather than to recurring, generic business cases on process level. The different notions of task patterns result from the different interpretations of tasks in the fields of BPM and HCI discussed in Section 4.1.

4.3 Generic Approach

User-driven process composition through integrated support in the actual end users' working applications is proposed in the context of evolving workflows [Her00]. The latter study suggests that preliminary process models should be constructed automatically based on the captured user interactions. Further work shows that attempts to enhance personal task management through contextual support and task (pattern) mining in a software system can introduce discrepancies between the user perception of how tasks should be organized and what the system offers [RC07]. Hence, automated task recognition approaches still need further research in order capture consistent task flow towards business process composition on personal and on enterprise level.

Furthermore, process composition approaches based on automatic recognition of users'

working tasks in a heterogeneous application environment basically result in process mining [vdAW03, vdAW05]. Such approaches hence do not allow end users to establish a direct relationship between their actions and the emerging processes. The lack of such relationship hinders the *"informed participation"* of end users in business process composition and does not foster *"social creativity"* [FGY+04] where domain experts proactively drive process optimization in enterprises. Therefore, the thesis suggests enabling end-user driven business process composition through explicit user interaction with light-weight, simplified task items pertaining to personal task management. This approach is considered relevant through a large body of research on user-centric process support [ADMG97, Ber00, Jor04, RRMvdA05, HRD+06].

The thesis fuses findings from related research discussed in the state of the art analysis in Chapter 3 and extends known user-centric approaches for task management and business process management towards end-user driven business process composition. Thereby the findings and elicited requirements from the empirical work (cf. Chapter 2) are considered. An overview of the generic approach is provided in Figure 4.1 and discussed in the following.

4.3.1 Personal Task Management

The thesis suggests enabling end users to structure their working tasks in light-weight, hierarchical to-do lists, and to delegate (sub-) tasks to other persons over email from their common working applications in the local workspace. These aspects are reflected respectively through steps (1) and (2) in Figure 4.1 and directly address R1 and R2 from the empirical studies. Steps (1) and (2) aim at providing a gentle slope of complexity [MCLM90] for process tailoring by end users. Concretely, end users without any interest in modeling or monitoring processes, i.e. workers according to [MCLM90], are enabled to manage to-do items in the local task list, without entering proprietary task management environments. This aspect is not considered in related work on user-centric process support, where no integration in the current users' task management applications is discussed [Ber00, Jor04, HMBR05, HRD+06]. On Figure 4.1 the personal workspaces of users $U_1 - U_4$ are shown as rectangle containers for the respective users in the top layer. The ovals represent user tasks, where e.g. task A has sub-tasks A_1 and A_2, task A_2 has sub-tasks $A_{2.1}$ to $A_{2.m}$ etc. The dotted line arrows represent task delegations over email, e.g. user U_1 has delegated task A_1 to users U_2 and U_3. Tasks B and C are thereby the tasks resulting from the delegation respectively in the personal workspaces of users U_2 and U_3. Recipients of delegated tasks are able to negotiate these tasks, to structure accepted tasks and to delegate resulting (sub-) tasks further. Thereby every end user structures their tasks based on the individual knowledge about how to best decompose the work items. Every user further manages their tasks according to their individual work practice. In that sense every user is shaping the resulting process facets in their area of expertise through managing their tasks in the personal workspace.

Programming by example [Cyp93, Lie01] towards business process composition is enabled through capturing user activities on explicit task representations in the task management system. Captured, user-defined task hierarchies and the underlying history of captured user activities such as task creation, structuring, editing and update of the processing state, provide examples of how the users have processed their personal tasks. These examples can be used to reconstruct the complete processing of an explicit task item in similar cases, thus providing reusable task and process knowledge for ad-hoc process support.

4.3.2 Task Delegation Graphs

To capture overall enterprise processes, it is further necessary to reconcile the individual task hierarchies of multiple process participants. This can be realized through tracking users' task management activities in the personal workspaces and the related email exchange for task delegation, and interconnecting related tasks to end-to-end *task delegation graphs* (cf. [SSS07]). The thesis defines a task delegation graph as follows.

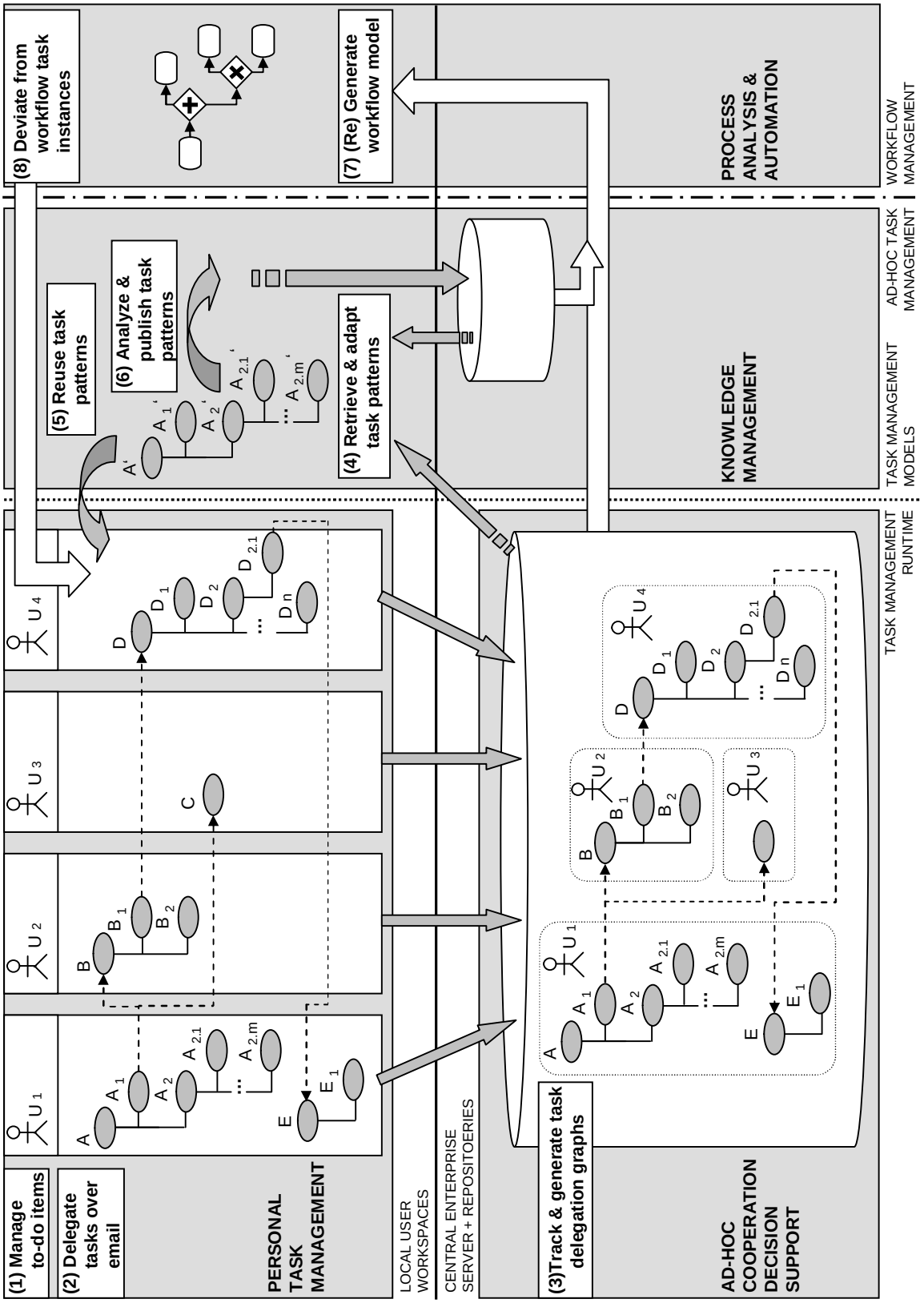


Figure 4.1: Generic approach for end-user driven business process composition

Definition 4.1: (Task Delegation Graph) A task delegation graph is a weakly-structured process model, which is composed of user-defined task hierarchies that are interconnected based on the task delegation flow. Formally, a task delegation graph is a tuple $S = (V, D, StrE, DlgE, DataE)$ where:

- V is a set of tasks, and D a set of data elements (artifacts)
- $StrE \subseteq V \times V$ is a structural relation according to the hierarchical decomposition
- $DlgE \subseteq V \times V$ is a structural relation according to the delegation flow
- $DataE \subseteq V \times D$ is a set of relations between tasks and artifacts

With respect to structural relationships between tasks, a task delegation graph is a tree [Die00] $G (V, E)$ with a set of vertices V and a set of edges E, where $E = E_h \cup E_d$ with $E_h \subseteq E$ being the set of edges representing hierarchical decomposition, i.e. connecting parent tasks with their respective sub-tasks, and $E_d \subseteq E$ being the set of edges representing delegations, i.e. connecting requester tasks with recipient tasks. An arbitrary sub-graph of a task delegation graph is also a tree. The term "graph" is used to leave room for concept extensions, e.g. for embedding cycles in user-defined task structures.

The tracking and interconnection of user-defined task hierarchies into task delegation graphs are denoted with step (3) in Figure 4.1. The above definition provides the associations for constructing a task delegation graph as shown in the central enterprise server layer of the task management runtime in Figure 4.1. The dotted-line areas represent user containers for users U_1 - U_4. These containers comprise the task hierarchies of each user that relate to the process initiated through task *A*. User associations are not explicitly provided in Definition 4.1 as these may be based on different user entities, denoting single users or generic user roles. The email exchange for task delegation is captured in *task delegation dialogs* which allow transparency and analysis of the collaborative flow on tasks. Further associations are enabled for task analysis, and for process formalization through workflow models. All associations are discussed later on in the task management model. The repository structure and related services that enable the aggregation of data for constructing task delegation graphs are discussed in detail in Chapter 7.

Unlike visual languages for modeling of collaborative processes [Swe93, GH98] task delegation graphs do not require any initial process model or preliminary knowledge of a process. They are themselves emerging process models which unfold during end users' task management activities and enable enhanced transparency in evolving collaborative processes (R3). For composing a task delegation graph, the end users are not required to interact with any visual notation but with light-weight, hierarchical to-do lists in the personal workspace. Hence, end users are composing weakly-structured process models through *"direct manipulation"* [Bla06] of explicit task instances in a task management system. Task delegation graphs are composed implicitly, in the background by capturing the task management activities of multiple participants, who are responsible for different process areas. In that sense, task delegation graphs are composed through "collaborative programming by example" [Lie01] on enterprise level and represent execution examples of weakly-structured process instances. A task delegation graph contains only the task context information and attributes of task instances, which are specified by end users in the personal task management environment. Thus, end users are not required to learn a visual language or formal notation to interact with a task delegation graph, but are rather enabled to view a different representation of the task instances that they have specified in the local workspace, including the tasks of all other involved stakeholders. Appropriate associations of resources and human actors' information are considered in task delegation graphs to capture the complete control flow, document flow, and user information in terms of task assignments.

Replication of task instances and related data to central repositories is considered in related

work for composition of enterprise processes that spread beyond the personal workspace [ADMG97, Ber00]. The major difference of task delegation graphs to such approaches is that a collaboratively handled task is available in both – in the requester and in the recipient task structures. For example, in Figure 4.1 task A_1 is available in the task list of user U_1 and represented through tasks *B* and *C* respectively in the task lists of users U_2 and U_3. Local task representations are considered important with respect to ensuring a gentle slope of complexity [MCLM90]. Concretely, through providing local task representations in the personal workspace of each user, the users are not required to engage with proprietary task management environments or a process overview in order to see the status of delegated tasks. Each user that is involved in a collaborative task is able to view the task status from their local task instance. Thereby the users who have delegated tasks to other persons, i.e. task requester, and the task recipient(s) can have different requirements towards the respective task representation in their local workspace. A requester expects a result and may be interested in progress information, or if a task has been accepted or rejected by the recipients. The recipient on the other hand delivers the result and may need to acquire an approval from the requester and to view the approval status. Such perspectives are not considered in related work on user-centric process support, where a single task instance is transferred between different parties or accessed in a shared manner [ADMG97, Ber00].

While the discussed approach enables end users to work in the personal workspace as usual, through task delegation graphs local developers or domain experts are able to view the overall process flow to determine work distribution by possibly identifying approaching bottlenecks and escalations. This provides decision support in evolving collaborative processes beyond the capabilities of common email and to-do list applications [Mül05]. End users, which need advanced transparency in the overall process flow can benefit from the overview provided by task delegation graphs. In that sense task delegation graphs aim at extending the end users' expertise with their common task management environment by providing added value for personal task management (cf. steps (1) and (2) in Figure 4.1). Through this, incentives are provided to end users to climb the *"tailorability mountain"* [MCLM90] and to gradually engage in process tailoring as informed participants [FGY+04].

4.3.3 Task Patterns

For enabling exchange, adaptation and reuse of process knowledge (R4), as well as tracing of relationships between running processes and best-practices (R5) and tracing of best-practice evolution (R6), the thesis suggests the usage of generic *task patterns* [RRMvdA05, RCKM06, OGR07, JGOR07, GOR+07, Sch09]. The thesis defines a task pattern as follows.

Definition 4.2: (Task Pattern) A task pattern is a reusable task structure, comprising one task with its sub-task hierarchy and the complete context information of the contained tasks such as name, description, used artifacts and involved persons. Formally, a task pattern is a tuple $S = (V, D, StrE, SugE, DataE)$ where:

- V is a set of tasks and D a set of data elements (artifacts)
- $StrE \subseteq V \times V$ is a structural relation according to hierarchical decomposition
- $SugE \subseteq V \times V$ is a relation based on suggested task pattern references
- $DataE \subseteq V \times D$ is a set of data associations between tasks and artifacts

With respect to structural relationships between tasks, a task pattern is a graph [Die00] $G (V, E)$ with a set of vertices V and a set of edges E, where $E = E_h \cup E_s$ with $E_h \subseteq E$ being the set of edges representing hierarchical decomposition, i.e. connecting parent tasks with their respective sub-tasks, and $E_s \subseteq E$ being the set of associations representing suggested task pattern references. The latter allow self references (cycles) in task patterns.

Task patterns abstract from case-specific information to provide a generic best-practice recommendation. Therefore they replace the delegation flow from task delegation graphs with suggestion references. The abstractions are discussed in detail in the task pattern model later on. Similarly to the definition of task delegation graphs (Definition 4.1), the task pattern definition does not provide user associations because task assignments and expertise recommendation can be based on concrete users or generic user roles. The human actor associations are discussed in the task pattern model further in this chapter along with the task pattern structure and attributes.

Task patterns serve as task models for ad-hoc process support as discussed in Section 4.2. The thesis suggests extraction of task patterns from executed processes as depicted in step (4) on Figure 4.1. Extracted task patterns represent process examples which are abstracted from a specific ad-hoc process instance, and are applicable for handling recurring cases in ad-hoc processes. Such task patterns contain recommendations about task decomposition, used resources, delegation flow and involved stakeholders and can be reused to reconstruct a task delegation graph according to a captured previous experience (cf. step (5) in Figure 4.1). In the sense programming by example [Cyp93, Lie01], i.e. capturing and repeated execution, of weakly structured process models is enabled. Reuse of task structures is considered relevant in related research on user centric-process support [Ber00, Jor04, HRD+06]. Definition of task patterns from scratch as explicit best-practices is also considered in the thesis.

Independently of the way in which task patterns are defined, these can be published for reuse in shared enterprise repositories (cf. [RRMvdA05]). This is depicted through step (6) in Figure 4.1. Steps (4) - (6) enable SER [FGY+04] of weakly-structured process models. The underlying repository structure and the different repository types are discussed in Chapter 7.

Task patterns can provide added value for personal task management through the possibility to reuse previous process knowledge and to compare running activities with related best-practices (task patterns) in a structured manner. With this respect task patterns are seen as another possibility to extend existing users' working experience and skills related to personal task management towards end-user driven business process composition.

4.3.4 Process Formalization

For enabling process tailoring as collaboration [MM00] towards automation of rigidly recurring processes through workflow engines (R7), the thesis suggest export of user-defined, weakly-structured task delegation graphs to structured workflow models as shown in step (7) in Figure 4.1. For workflow export the captured task instance data from the runtime repository is considered, as task instances represent steps in concrete process executions and enable analysis of the associated control, data and event flow and human actors' information in terms of task assignments. Task delegation graphs and local task hierarchies in the users' to-do lists incorporate structural information and can display the current processing status of tasks. However, they do not provide information about the actual process flow, i.e. the sequence in which the tasks have been performed. This information is provided in the derived formal workflows. Derivation is based on the structural information of captured task delegation graphs, i.e. ad-hoc process execution examples, and associated task change history. Thus, the value of programming by example [Cyp93, Lie01] of weakly-structured process models is extended towards programming by example of structured workflow models. The structured workflow models provide a different representation of captured processes, which complies with a given formal notation. These models can be used for process analysis and automation and can provide further added value for the management of enterprise operations beyond the level of personal task management.

Task patterns are not considered directly during process formalization as they hold task model data and do not incorporate execution details. Nevertheless, task patterns can be used to trace similar cases based on the reuse history and to precise the derived formal workflows, i.e. through evaluating all ad-hoc process instances that have resulted from reuse of the same task pattern.

Therefore the task pattern repository is linked into the transition from task delegation graphs to workflow models in Figure 4.1.

Process tailoring by end users can enable enterprises to adapt to changes in business processes without a need for extensive and time-consuming external consulting or software development. The thesis suggests that after a formal workflow model has been derived, end users without technical expertise should be further able to extend it on demand in the context of use (cf. [WJ04]). To enable such extensions the ad-hoc task management system and the workflow management system are interconnected and allow deviations from workflow task instances with ad-hoc tasks as shown in step (8) in Figure 4.1. Captured deviations are considered when the workflow model is redesigned. The deviating task structures are embedded into the original workflow model, thus extending it with custom, user-defined tasks. Steps (7) and (8) in Figure 4.1 enable SER [FGY+04] in the context of structured workflow models.

The basic interrelation between ad-hoc task instances and structured workflows is introduced in this chapter. The method for transformation of weakly-structured to structured process models, and for extending workflow models based on ad-hoc deviations is presented in Chapter 6.

4.4 Runtime Task Management Model

The runtime task management model defines the task management model at instance level, i.e. it describes the underlying concepts and relations for ad-hoc task management through explicit task representations in light-weight to-do lists in the course of running ad-hoc processes. The runtime task management model aims to leverage user experience with standard office applications for task management (to-do lists) and collaboration (email) towards the composition of business process models. Thereby the model considers the major requirements for end-user driven business process composition from the preliminary empirical studies (cf. Table 3.1). The runtime task management model is shown in Figure 4.2 and discussed in the following.

4.4.1 Task Instances

The basic approach underlying the presented task management model is to involve end users in business process composition through enhanced personal task management. In order to achieve a gentle slope of complexity [MCLM90] users are not required to model processes in upfront modeling environments, but to declare and structure work items related to their day-to-day activities in light-weight, personal to-do lists (R1). Studies in this direction [Ber00, HRD+06] exemplify the need for enhanced context provision environment, where users can declare tasks to different details' level and enrich them with other important information, i.e. through textual descriptions or through attaching related documents. The model entities related to task instances are discussed in the following sections. Artifact and user entities are relevant for both – task instances and patterns and are discussed in the end of the chapter.

4.4.1.1 Task Instance Structure

A **root task** association is considered for ad-hoc task instances. A root task is the generic task that identifies the business goal of an overall process. Each root task identifies a distinct ad-hoc business process, and each ad-hoc business process is identified through one and only one root task. Thus in the thesis, the term "root task" is used as a synonym to ad-hoc business process.

Parent/sub-task relationships are further considered for hierarchical structuring of task instances. Each task in the to-do list can contain an arbitrary number of sub-tasks. The hierarchical task decomposition enables end users to purposefully declare and structure work items in personal to-do lists (R1). The usage of hierarchical task decomposition is suggested by a large body of research on user-centric process support [Ber00, Jor04, HRD+06, GOR+07].

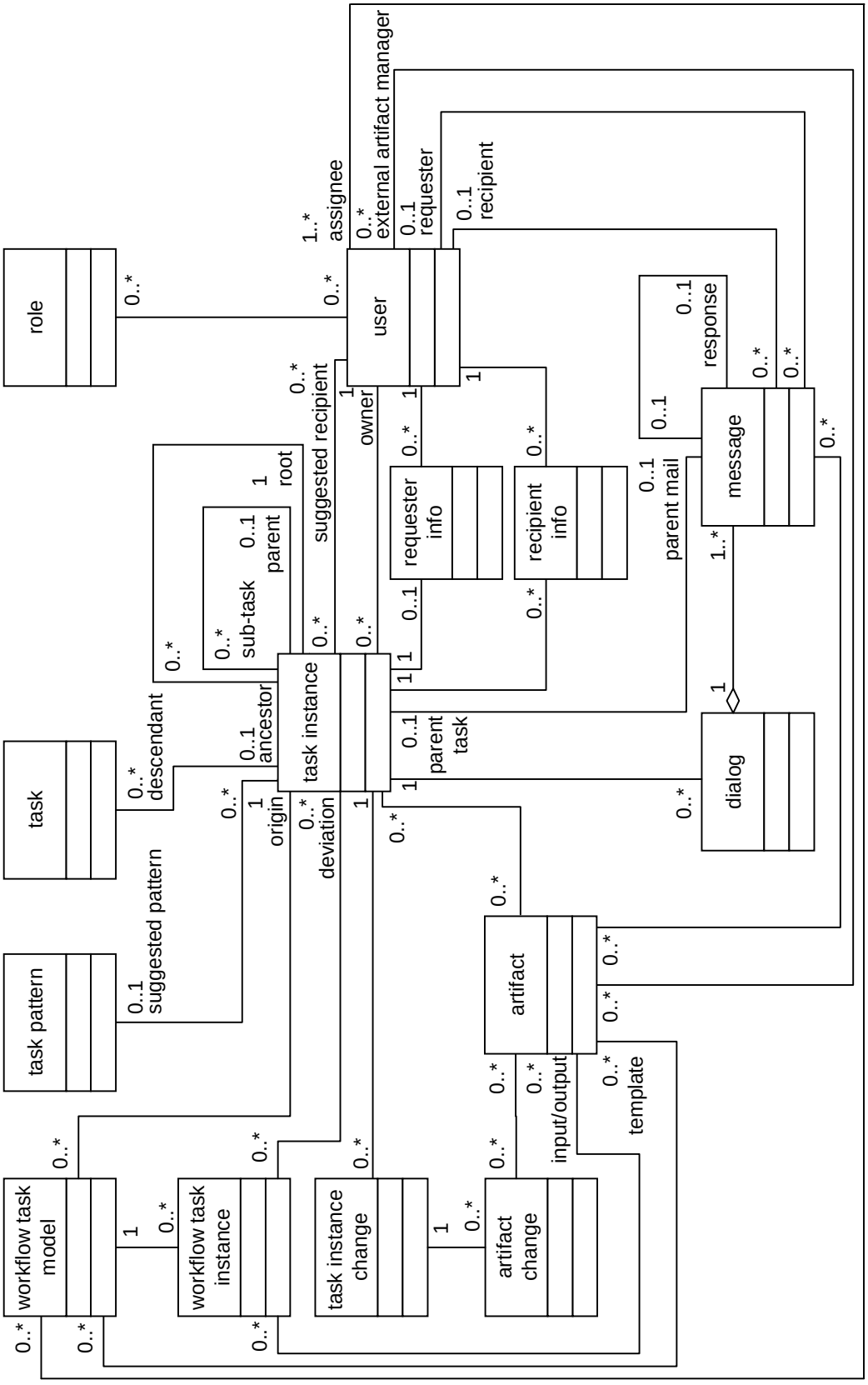


Figure 4.2: Runtime task management model

The complexity of an ad-hoc process cannot be determined in advance, as such processes are considered to be initially ill-defined [RRMvdA05, HRD+06]. Hence, a process can remain simple, defined in the personal workspace through a root task with several sub-tasks. For example, a root task "organize workshop" can involve sub-tasks "reserve a room", "order meal", "book hotel for visitors", where "order meal" can involve further sub-tasks such as "find catering company", "compare prizes" etc. The task decomposition depends on the user's attitude and the level of specificity which they wish to achieve. If involvement of multiple persons is necessary, the resulting collaborative process can comprise task hierarchies of more than one user, e.g. the "order meal" task can be delegated and performed by other person than the one working on the "organize workshop" task. In the latter case, the tasks of both users will have a root task association to the same task "organize workshop". Thus, the root task relationship is especially important for capturing collaborative processes that evolve in the workspaces of various users. This relationship enables associating task hierarchies from the various workspaces to a common overall process. Related task management models [GOR+07] do not consider such relationship.

4.4.1.2 Task Instance Analysis in the Context of SER

Ancestor/descendant relationships are proposed in the task management model to support task analysis in the context of SER (cf. Table 3.1). Ancestors and descendants for a given task instance or a task pattern can be both – task instances or task patterns. For example, if a given task pattern *A* has been applied in an ad-hoc process, and has resulted in a task instance *A'*, then *A* is considered as the ancestor of *A'*, and *A'* is a descendant of *A*. The establishing and effects of ancestor/descendant relationships are discussed in details in the method for composition of weakly-structured process models in Chapter 5.

Each ancestor can have further ancestors, and each descendant can have further descendants if reuse has been performed iteratively. While the ancestor references enable backward tracing of the origin of a given task, the descendant references enable forward tracing of task reuse in further cases. If a task has been reused multiple times it has multiple descendants. Ancestor relationships are considered in related work, i.e. as *"task source relationships"* [GOR+07]. However, the latter study does not consider forward tracing of evolving task hierarchies through descendant relationships.

4.4.1.3 Recommendation and Guidance Based on Suggestions

Suggested task pattern associations to a given task pattern are further proposed by the thesis. Suggested task patterns enable knowledge distribution and enhanced guidance according to an available best-practice. The establishing and use of suggestions are discussed in Chapter 5.

4.4.1.4 Task Instance Attributes

Task items in the to-do list contain context attributes in order to respond to the requirements for personal task management that manifested in the empirical studies (cf. Table 2.13). The task instance attributes are shown in Figure 4.3 and discussed in the following.

The **identifier** attribute provides a unique identification of a task instance throughout the task management system. Identification is required as composition of business processes may require distribution of task instance information on different system entities beyond the personal workspace, and interconnection between task instances, artifacts and human actors. To enable such interrelations an identification of the different entities is needed. The distribution of the different entities is discussed in more detail in Chapter 7.

The **type** attribute can have three different values: *root task*, *accepted task* and *sub-task*. The type identifies the structural task relationships in task delegation graphs:

- A *root task* has no parent tasks and there are no previous tasks in the task hierarchy in the local workspace or in the overall collaborative process, i.e. a root task is the first task in

an end-to-end task delegation graph (cf. task *A* in Figure 4.1).

- An *accepted task* emerges when a user accepts a task request. An accepted task appears on root level in the local, personal task list of a recipient. However, in the context of the overall collaborative process this task is subsequent to previous tasks, i.e. to the requester task (see tasks *B, C, D* and *E* in Figure 4.1).
- A *sub-task* has a parent tasks, i.e. it does not reside on root level in the task hierarchy within the local to-do list or within the overall collaborative process.

The **index** attribute specifies a (zero-based) index of a sub-task in the sub-tasks' collection of a parent task. Hence, root tasks and accepted tasks have an index 0. Unlike the other attributes, the identifier, type and index attributes relate to system information for task instances in overall business processes and may not be exposed to users in the task management application.

The **name** and **description** attributes of a task instance are self-explanatory and provide human-readable task information.

The **start date** and **due date** attributes address $R_{q\,2.1}$, $R_{q\,2.2}$ and $R_{q\,2.3}$ (cf. Table 2.13), i.e. these attributes enable: setting of time constraints for tasks, reminders for due dates, task order filtering for showing tasks in the order of their due dates, and estimation of task completion times.

The **priority attribute** aims at enabling task ordering according to the task priority with respect to $R_{q\,2.2}$ (cf. Table 2.13).

The **status** and **percent complete** attributes enable users to monitor the progress of their tasks, locally and in a collaborative setting according to $R_{q\ 3.2}$ (cf. Table 2.13). Similar attributes are suggested for task instances also in related task management models [GOR+07]. However, the latter study does not discuss states related to the collaborative handling of tasks, which is an important aspect of the runtime task management model presented in this thesis. The states of a task instance are discussed in the next section. In the thesis statuses are considered as human-readable values for the task states that are represented (visually) in the task management system.

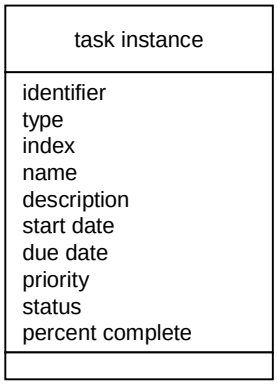


Figure 4.3: Attributes overview of the user entity

4.4.1.5 Task Instance States

Task instance states are changed and updated through user actions on task items in a to-do list. The states are reflected through corresponding status indications for task instances. The runtime task management model supports collaborative task handling through the basic states shown in Figure 4.4. Further state transitions are discussed in the method for composition of weakly-structured process models in Chapter 5.

New is an initial state, in which a task instance is set after the user has created it explicitly or by accepting a task request. This state is considered also in [GOR+07]. The corresponding status attribute for that state is *not started.*

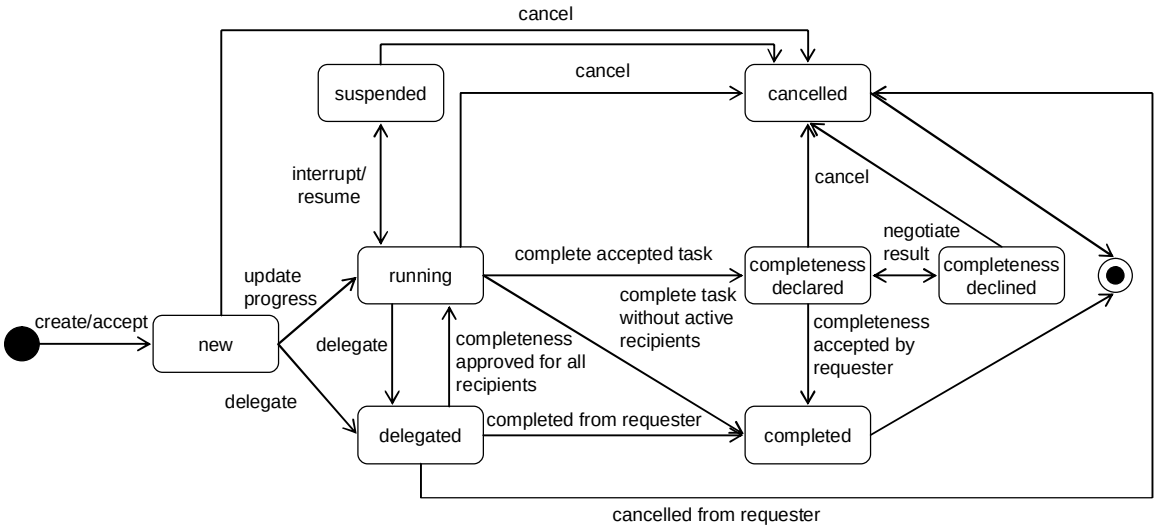


Figure 4.4: Task instance states

Running state denotes that certain actions are being performed to accomplish the task. This state is also considered in [GOR+07]. A task instance is set in this state upon changes to the percent complete, i.e. setting this between 0 and 100, or when the corresponding status attribute (*in progress*) is explicitly set for a task instance.

If a parent task that is in state *new* is set to state *running*, the latter state is not applied to the sub-tasks. The state change is restricted to the parent task as depending on the task definition and decomposition the parent task may denote certain activities that are not reflected through explicit sub-tasks. Such activities may be performed as preparation for the execution of the sub-tasks. Furthermore, sub-tasks may need to be executed in sequence, i.e. reach the running state subsequently. Thus the state of each sub-task needs to be adjusted individually.

On the other hand, if a parent task is in state *new*, setting a sub-task to state *running* sets also the parent task to that state. This handling considers that a parent task defines the overall goal that should be reached by the sub-tasks. Such semantics are considered in a large body of research with respect to task decomposition [JK06a, JK06b, Joh03, Ber00, Jor04, HRD+06, GOR+07]. Thus if a sub-task is running, then it is running as a part of the overall parent task and the latter is also considered as running.

Suspended state denotes that a task has been started but put on hold. The user is able to explicitly set a task instance in this state through adjusting the status attribute (to *suspended*). This state is considered also in [GOR+07]. The relationships between the *running* and *suspended* states are analogous to those between the *running* and the *new* states explained above.

A *running* parent task can have *suspended* sub-tasks. In this case the parent task itself denotes running activities for which no explicit sub-tasks are created or there are be other running sub-tasks of the parent task. Thus, suspended sub-tasks refer to suspended partial activities towards achieving the overall goal of the parent task, which does not exclude other running partial activities (sub-tasks).

On the other hand, the thesis suggests that it should not be allowed to have a *suspended* parent task with a *running* sub-task. This rule results from the semantics of hierarchical decomposition discussed for the running state, i.e. if a sub-task is running, then it is running to achieve the overall goal of the parent task and the latter is also considered as running. Setting tasks to state suspended and resuming tasks by moving them from the suspended to the running state have implications when the tasks of different users in a task delegation graph are affected. These state transitions are discussed in more detail in Chapter 5.

Completed and **cancelled** states that are shown in Figure 4.4 are generic and correspond to corresponding states in related work [GOR+07] where the state *cancelled* occurs as *terminated.* In Figure 4.4 only the normal state transitions related to completion and cancellation are depicted for simplicity. The thesis suggests that in order to preserve flexibility in ad-hoc processes, completion and cancellation states should be reachable from all other states, i.e. it should be possible to cancel or complete any task that is not already cancelled or completed. With this respect the thesis considers complex implications from completion and cancellation that affect tasks of multiple users throughout a task delegation graph. These implications and the related intermediate cancellation and completion states are discussed in details in Chapter 5.

A state *finalized* is considered in [GOR+07], which is reached after task completion or cancellation are approved. This state is relevant for managing a single, shared-accessible task instance by users with different task roles [GOR+07]. However, this state does not sufficiently address the paradigm of task delegation graphs, where the states of requester and recipient tasks need to be considered simultaneously. Unlike [GOR+07] the task state model proposed in this thesis considers additional task states related to the collaborative handling of tasks.

Delegated is a state, in which a task instance at requester site is set after it is delegated to other persons. This state is reflected with status indication *waiting for someone else* and denotes that there are other stakeholders involved in that task. In the task delegation graph in Figure 4.1 this is a relevant state for tasks A_1, B_1, and $D_{2.1}$. Related work on collaborative, ad-hoc process support [Dus04] considers a delegated state with different semantics, denoting that the responsibility of the task has been shifted completely to the recipients of the delegation. Unlike that, the delegated state introduced in the thesis considers a shared responsibility between requester and recipient, where the requester is further able to perform on the task. Concretely, in a task delegation graph requester and recipients have different task representations of the delegated task in the local workspaces, which allow each of them to further work on that task by adapting the local task representation and through this also the overall process (cf. Section 4.3.2).

Completeness declared state is reached when a user marks an accepted task as completed. In this case the task is not transferred automatically to state *completed* but a completion declaration is issued to the task requester and the status attribute of the recipient's task is changed accordingly to *completeness declared*. The intermediate completeness declared state and the handling of the completion declaration allow negotiation of deliverables where the task requester can ensure that the delivered result matches the initial assignment. The collaborative task handling is discussed in details in Chapter 5.

Completeness declined state is set to a recipient's task, when the corresponding completeness declaration is rejected by the task requester. The status attribute of the recipient's task is set to *completeness declined* to keep the recipient aware that the delivered result is insufficient and the task needs to be reworked. In the task delegation graph in Figure 4.1 this is a relevant state for tasks *B*, *C*, *D*, and *E*. Tasks with this state can be again declared as completed (after a correction of the results). This process runs iteratively until a satisfactory result is achieved. Approval of a completion declaration sets the recipient task in state *completed*.

When completeness declarations have been returned by all task recipients and accepted by the requester, the original requester task receives a completion rate of 99% and is set to state *running*. The requester can then perform concluding activities related to the previously delegated task before setting it in a final *completed* state.

The intermediate completeness declared and completeness declined states are not considered in related work on ad-hoc process support [GOR+07, Dus04]. These states relate to basic speech acts for exchange of tasks and deliverables (cf. Section 4.4.3) and support the collaborative handling of tasks in task delegation graphs.

Transferred intermediate sub-state is additionally considered for completely transferring a task instance in its current state to another user. Transfer reassigns a task by preserving its structure and position in the overall task delegation graph but substituting the responsible person,

i.e. moving the transferred task to the to-do list in the local workspace of another user. Transfer is considered for cases when a user is no longer able to process a task and should be substituted with another user, who takes over the job in its current state. A transfer can be triggered to only one person at a time and only on root tasks or accepted tasks. The transferred sub-state can occur in combination with any of the other states and is not shown in Figure 4.4 for simplicity. A task is set in the *transferred* sub-state when a procedure for transferring the task to another user is triggered. The procedure may be collaborative or administrative. In collaborative transfer, the user that is currently responsible for the task, i.e. who has the task in their to-do list, triggers a transfer by sending a transfer request to another person as discussed further in this chapter. In administrative transfer, the task is reassigned by an administrator, who changes the task assignments on the server. Administrative transfer may be required if the person that is currently responsible for the task is not available for collaborative transfer. A transferred task is loaded from the server to the to-do list of the new assignee. All relationships to dependent collaborative tasks are updated for the new assignee and supported with notifications as discussed in Chapter 5. When the transfer procedure is completed the *transferred* sub-state is removed.

4.4.2 Task Instance Changes

Task instances can undergo various changes. These changes reflect purposeful user actions on task instances declared in a to-do list. Such task changes are considered important for business process composition in the sense of programming by example [Cyp93, Lie01], because they represent explicit task processing stages outlining the overall process flow. That is to say, changes to task instances in running processes denote actions in the system that provide an example about the actual execution of the user tasks in the course of an informal business process. For example, when performing programming by example of macros, a user captures their actions in the respective system in the form of macro recordings. The recording in the context of end-user driven business process composition comprises the task change history of all task instances that have been managed during the execution of an informal business process.

In order to capture the complete task change information, the following task change types are considered: *task created, task moved, task removed* and *task context changed*. The task instance change entity attributes are given in Figure 4.5 and explained in the context of the different change types in the following.

An **identifier** attribute is considered for unique identification of a task change throughout the system. A value for this attribute is set for all task changes independently from their type. Such identification is needed as changes can be decoupled from the corresponding task instances and manipulated as independent objects to enable task instance recovery.

The **time** attribute specifies the exact time when the change occurred. All task changes receive a value for this attribute independently from their type. Time is not considered reliable for uniquely identifying a task change as changes can occur simultaneously depending on the underlying implementation.

The **type** attribute denotes one of the task instance change types specified above. Each task instance change has a value for this attribute. The first three change types given above: *task created, task moved, task removed* enable detection of changes in the overall task structure. These change types are necessary as the hierarchical task decomposition provides only information about the current task structure but not about changes in this structure. Changes that occur through adding or shifting of tasks in the course of running processes can denote exceptional behavior or compensation handling due to some unplanned circumstances.

Task created changes receive values for all attributes that are specified in the created task instance and additionally receive the identifiers of root and parent task instances. These changes enable full recovery of the created task instance, even if the task instance is removed from a user's workspace or from an overall process. The recovery is enabled by decoupling the entity data from the actual entity instances, i.e. root and parent task instances are not associated to the

task instance change entity in the way that these are associated to the task instance (cf. Figure 4.2). Instead, only the identifiers of root and parent task instances are contained in the task change entity, whereas the full contents of the respective root and parent task instances can be recovered from their corresponding *task created* changes. This recovery is necessary in order to capture the complete example of how a process was structured and managed even if some of the task instances have been removed from the task management system during or after process execution. Such recovery is not considered in related task management models [GOR+07].

Task moved changes assign values to the: (i) *root task identifier* - if a task has been moved to another process, i.e. root task (cf. Section 4.4.1.1); (ii) *parent task identifier* – if the task instance has been moved to another parent task; (iii) *index* – if the index of the task in a parent task's sub-tasks collection has changed.

Task removed changes assign values to the *task identifier* attribute. The place in the task hierarchy, from which the task has been removed, can be determined from previous task changes.

Task context changes refer to changes in the values of task attributes that are visible and used by end users for task management such as: *name*, *description*, *start date*, *due date*, *priority*, *status* and *percent complete*, and task instance associations with artifacts (cf. Figure 4.2). Hence, task changes from the *task context change* type receive values to all changed attributes of task instances, and references to one or more artifact changes where applicable.

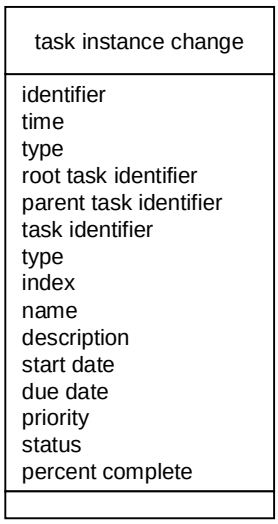


Figure 4.5: Task instance change attributes

4.4.3 Exchange of Tasks and Deliverables

A significant body of research discusses that collaborative processes are influenced by the social nature of human work. [Win86] proposes a *"language/action perspective"* for designing cooperative work. This perspective is leveraged in [FGHW88] where organizational interaction is designed as a *"network of negotiated commitments"* based on speech acts. Further studies analyze offices as *"systems of communicative action"* and apply a speech act based approach for modeling office information systems [ALL88]. Such studies strongly support the use of speech act techniques for the exchange of tasks and deliverables in the context of collaborative task management. Speech acts are further seen as related events that *"participate in larger conversational structures"* [Win86]. Thus, for enabling email-integrated exchange of tasks and deliverables (R2), the runtime task management model utilizes two generic entities – messages and dialogs.

4.4.3.1 Messages

Messages are emails, which have generic types that pertain to basic speech acts. The actual discourse during message exchange takes place in the email text. This allows open-ended collaboration and prevents from submitting user behavior to strict speech act rules, which is a known limitation in speech acts' adoption [But94]. Each message can have one or more associated artifacts (cf. Figure 4.2 and Section 4.6), which are included in the message as common attachments.

Messages are exchanged for the following collaborative operations: (i) *task delegation*, (ii) *consolidation of operations that affect multiple collaborative tasks throughout a task delegation graph* (e.g. task cancellation and completion), and (iii) *collaborative transfer of a task to a different user*. The various messages are discussed in the following whereas the complete collaborative handling for task delegation is discussed in Chapter 5.

4.4.3.1.1 Messages for Task Delegation

The messages for task delegation are: *request, request negotiation, request acceptance, request declination, request termination, completion declaration, acceptance of completion declaration,* and *declination of completion declaration.*

A task *request* message is sent for a given task in the personal to-do list of a system user. It is the first message in a *"conversation for action"* [Win86, FGHW88]. The request message receives a *parent task* association to the delegated task, and *requester* and *recipient* associations respectively to the sender and the recipient of the request message (cf. Figure 4.2). If a request for task delegation is sent to multiple recipients, these are split so that a single (replicate) message is sent to each recipient individually. The split allows managing a separate dialog for each recipient as discussed further in this chapter.

Request acceptance and *request declination* are basic speech acts in a *"conversation for action"* [Win86, FGHW88]. Corresponding messages are known also from common functionalities in office tools, such as e.g. meeting requests in Microsoft Outlook. *Request acceptance* produces a task in a recipient's to-do list. This task receives a *parent mail* association to the accepted request message (cf. Figure 4.2). This association points at the message, from which a task has resulted. This message can be also the last negotiation message for a given task request that is sent by the requester.

Request negotiation allows users to discuss and change the boundary conditions of a task assignment. Negotiation of task requests is considered also in related task management models [GOR+07].

Request termination is an additional message type, which allows the requester to explicitly declare that a request is no more relevant, i.e. to recipient(s) who have not accepted or declined that request yet. Request termination can result for example if during negotiation the requester decides that the request is not feasible and wants to inform the recipient(s) for that final decision.

Completion declarations are an additional aspect of the task management model presented in the thesis which is not discussed in related task management models [GOR+07]. Completion declarations consider that task requests are interpreted by the different users as having *"conditions of satisfaction"* [Win86]. These conditions may be interpreted differently by the requester and the recipient of a task. The handling (send, accept, reject) of completion declarations enables consolidation of the requester's and recipient's satisfaction criteria.

All message types except a task request are considered as *responses* to previous messages. For example, request acceptance/negotiation/declination is a response to the corresponding request message. Iterative negotiations can be triggered, where each negotiation message is a response to the previous negotiation message. If a request termination is triggered, the termination message is considered as arising in response to the issued request message. The completion declaration is as well considered as a response to the request acceptance message. The *response* relationship is

depicted through a self-reference in the message entity in Figure 4.2. The complete information about a requester, requester task, recipient and recipient task in a delegation can be retrieved over the message references. In the local workspace of end users this information is available and transferred between different messages through embedded message attributes as discussed in the method for composition of weakly-structured process models in Chapter 5.

4.4.3.1.2 Messages for Consolidation of Operations with Global Impact

The messages for consolidation of operations with global impact on collaborative tasks are: *<operation> request*, *<operation> request negotiation*, *<operation> request approval*, *<operation> request declination*, and *<operation> request termination*, where *<operation>* can be one of: *cancellation*, *completion*, *suspension*, or *resumption*. These operations are discussed in detail in the method for composition of weakly-structured process models in Section 5.1.5. The provided message types pertain to basic speech acts and have analogous semantics to the corresponding message types for task delegation.

An *operation request* receives a *parent task* association to the task on which the operation is initiated, and *requester* and *recipient* associations respectively to the sender and recipient of the message (cf. Figure 4.2). If an *operation request* is sent to multiple recipients, these are split analogously to task delegation requests. *Recipients* for an *operation request* are determined through the server by evaluating tasks that are affected by a given global operation as discussed in the method for composition of weakly-structured process models in Section 5.1.5. All other message types for a global operation are associated as *responses* to the corresponding previous message (cf. Figure 4.2).

4.4.3.1.3 Messages for Collaborative Task Transfer

The messages for collaborative task transfer are: *transfer request*, *transfer request negotiation*, *transfer request acceptance*, *transfer request declination*, and *transfer request termination*. In contrast to delegations, task transfer does not require exchange of deliverables. The request handling in collaborative task transfer is analogous to that for task delegation. The target of the agreement is the complete transfer of a given task to a new assignee.

A transfer request receives a *parent task* association to the task for which the request is issued, a *requester* and a *recipient* association correspondingly to the sender and recipient of the message (cf. Figure 4.2). Recall that a transfer can be triggered to only one recipient at a time and only on root tasks or accepted tasks, thus no split of recipients is necessary. All message types other than *transfer request* are associated as *responses* to the corresponding previous transfer message (cf. Figure 4.2). When the requester of the transfer receives an acceptance for their request, they can trigger the transfer of the respective task in their to-do list.

4.4.3.2 Dialogs

A dialog (cf. Figure 4.2) represents a *"larger conversational structure"* [Win86] which is composed of multiple speech acts (messages) related to the collaborative handling of a given task between one *requester* and one *recipient*. A dialog is always initiated through a *request* message, considering a request as the first speech act in a *"conversation for action"* [FGHW88]. A dialog is determined through the associated *parent task*, and *recipient* of the request. Recall that if a task request is issued to multiple recipients, these are split and an individual request is sent to each recipient. Thus a single dialog is maintained between the requester and each recipient of a request message. The dialogs for different collaborative operations are discussed briefly in the following.

A **dialog for task delegation** is initiated through a *task request* and comprises the request and all subsequent messages related to the handling of the request including request acceptance, declination, negotiation, and termination. The dialog further comprises the messages related to the completion declaration, i.e. to the exchange and consolidation of task deliverables.

A **dialog for consolidation of an operation with global impact** is initiated through an *operation request* message and comprises all subsequent messages related to the consolidation of this operation between the requester and a given recipient. The recipients for the request are determined based on the tasks that are affected by the operation as discussed in Section 5.1.5.

A **dialog for collaborative task transfer** is initiated with a *transfer request* message and comprises all transfer-related messages. Collaborative task transfer can be performed only to a single person, thus no split of recipients is considered in this collaborative operation.

Dialogs for different collaborative operations may coincide if the request messages that initiate them are associated to the same *parent task* and *recipient*. For example, a dialog for consolidation of a global operation may coincide with a dialog for task delegation if there is a task *delegation request* message with the same *parent task* and a *recipient* association as the *operation request* message. In that case a single dialog is maintained which encompasses the agreements on the exchange of the task and the deliverables and all agreements on intermediate operations affecting the collaborative task.

To sum up, messages and dialogs enable enhanced transparency in evolving ad-hoc processes (R3) by complementing the structural overview of task delegation graphs with overview of collaboration and agreements on tasks. This transparency can help to reduce the search effort for task-related emails [BDHS03], and facilitate ad-hoc task and process management in email-centric task management environments.

4.4.3.3 Awareness for Collaborative Tasks

A central concept in the introduced approach for end-user driven business process composition is to enable a gentle slope of complexity [MCLM90] by allowing end users to gradually extend their skills with conventional task management (to-do list) and collaboration (email) applications. The task management model addresses a gentle slope of complexity through providing information about the further handling of collaborative tasks in the local workspace of end users. This information enables users to view the status of delegated tasks from the local workspace without entering a proprietary process overview. Indications about further delegations or transfer of collaborative tasks can provide incentives to end users to enter the process overview in order to view how a delegated task has evolved further and what other stakeholders have been involved. The local awareness for collaborative tasks is provided through the *requester info* and *recipient info* entities of the task management model (cf. Figure 4.2).

Requester info entities are relevant for accepted tasks. These entities contain information about the task requester depicted through the respective user association in Figure 4.2. Through the requester info, a task recipient is able to see who has requested a given task. Indications of the overall status and completeness percents of a requested task may be further provided through appropriate attributes. Awareness entities are managed through notifications. Basic notifications are discussed in Chapter 5.

Recipient info entities are relevant for delegated tasks and are managed in the local workspace of a task requester. These entities contain information about the task recipients depicted through the respective user association in Figure 4.2. Recipient info entities support collaborative task handling through a recipient status for a delegated task, which is maintained for each recipient based on the current state of the task delegation dialog. Following the message types discussed in Section 4.4.3.1.1, the following recipient statuses are considered for task instances: *requested, request accepted, request negotiated, request declined, request terminated, completeness declared, completeness approved, completeness declined.* The recipient statuses help a requester to get a quick overview of the collaborative status of tasks, without leaving the personal workspace. It is further possible to embed recipient's task information such as percent complete and task status in the recipient info. Thereby a status that indicates further delegation (i.e. *waiting for someone else*) may provide incentives to the requester to enter the process overview in order to follow how the process has evolved further.

4.4.3.4 Guidance for Task Delegation

Suggested recipients associations for task instances (cf. Figure 4.2) are inherited from task patterns after reuse. These recipients are automatically suggested if a delegation for the given task instance is triggered. Extraction, adaptation and reuse of task patterns are discussed in Chapter 5. Here it is important to stress that suggested recipients recommend expertise but do not point at persons that actually participate in the process. The suggested recipients can be involved in the process through explicitly sending task requests.

4.4.4 Workflow Tasks

Workflows refer to the procedural automation of business processes and need an explicit process definition to operate [Hol95, vdAvH02, Wes07]. In related literature [vdAvH02] the term "workflow" is used as a synonym for "business process". Similarly, the thesis uses the term "workflow" as a synonym for an "operational business process" which can be automated on a workflow engine. Thereby a workflow model is a formal model of an operational business process, which is composed of workflow task models (cf. Definition 1.4 and 1.5). In contrast to a task pattern, which provides a user-defined task model for an ad-hoc task, a workflow task model provides a model for a workflow task, i.e. for a task in an operational business process. A workflow task is considered as a generic task in a workflow (cf. Definition 1.2) and its attributes depend on the concrete Workflow Management System (WfMS) used for process automation. Workflow tasks are discussed in the thesis with respect to process automation (R7) by considering both, workflow task models as well as workflow task instances. The basic approach suggested in the thesis is to enable derivation of workflow task models from user-defined, ad-hoc task instances.

4.4.4.1 Workflow Task Models

A workflow task model, also referred to as workflow *task node* in the context of workflow graph definitions, is a task definition within a workflow model definition (cf. Definition 1.4 and 1.5). Following the idea for end-user driven business process composition, workflow models, and in particularly workflow task models, are derived from user-defined, ad-hoc task instances. Task patterns are not considered directly in the derivation of formal workflow models, as they do not reflect concrete process executions where temporal relationships between ad-hoc tasks can be detected. Task patterns can deliver different process execution variations in order to refine the derived sequence flow. Derivation of formal workflows is discussed in details in Chapter 6.

A task delegation graph can be transformed multiple times, producing different workflow models. Thus, an ad-hoc task instance can be used to derive multiple workflow task models (cf. Figure 4.2). Although different ad-hoc task instances can exist for the same case, i.e. these can be created from the same task pattern, one workflow task model has finally only one ad-hoc task *origin* (cf. Figure 4.2). This is the ad-hoc task instance, from which the workflow task model is generated and from which it inherits the context information such as name and description, required documents or document templates, and assignment to a human actor. Other related ad-hoc task instances can be retrieved through ancestor/descendant relationships to complement the workflow task model information.

Workflow task models do not refer to a concrete workflow execution. Thus, with respect to associated document flow, workflow task models can use artifacts from ad-hoc task instances as templates (cf. Figure 4.2), i.e. as documents that do not depend on a given execution context. The transformation of artifact associations from ad-hoc tasks to workflow task models is discussed in Chapter 6.

Workflow task models can further specify task assignments. Such assignments can be based on roles. In conventional visual process modeling notations like BPMN [OMG06] role-based assignments are widely used and represented through "swimming lanes". The presented task

management model relies only on user data that is available in email and to-do list applications (cf. Section 4.7) and does not discuss domain-specific roles. A workflow task model inherits the user assignment from its originating ad-hoc task instance. Multiple assignments can be included through merging collaborative tasks as discussed in Chapter 6. Thus, each resulting workflow task model has one or more assignees (cf. Figure 4.2). The assignments can be generalized through mapping user information to generic business or security roles according to the concrete workflow management system that is used for process automation. The transformation of task assignments is discussed in details in Chapter 6.

4.4.4.2 Workflow Task Instances

When a derived workflow model is instantiated, the contained workflow task models produce workflow task instances. Each model can produce multiple instances (cf. Figure 4.2). The workflow task instances inherit the static template artifact associations and the assignments from the workflow task model. These associations of workflow task instances are not shown in Figure 4.2 for simplicity. Additionally, as workflow task instances refer to concrete processes, they can use various artifacts as input and output. These artifacts can be produced and modified in the course of the given process instance.

The third challenge for BPM systems stated in the beginning of the thesis (cf. Section 1.2.3) is to enable adaptive BPM through user-tailorable process definitions. To address this challenge, the introduced task management model considers possibilities for extending formal workflow with user-defined hierarchies of ad-hoc tasks on demand. Ad-hoc tasks can serve as extensions, handling some steps that have been missed in the initial workflow definition. Hence, each workflow task instance can have one or more ad-hoc task instances as deviations (cf. Figure 4.2). This interrelation extends the SER [FGY+04] capabilities for the domain of operational business processes. The deviation handling is discussed in detail in Chapter 6.

4.5 Task Pattern Model

In the thesis a task pattern is considered as a reusable task structure, comprising one task with its sub-task hierarchy and the complete context information of the contained tasks such as name, description, used artifacts or involved persons (cf. Definition 4.2). A task pattern provides explicit best-practice recommendation for handling of recurring cases as introduced in [RRMvdA05, GOR+07] and clearly refers to the case dimension of business processes [vdABV+99]. Task patterns aim at providing enhanced capabilities for exchange, adaptation and reuse of process knowledge (R4) and serve as models for ad-hoc task instances in informal business processes. Task patterns, which have been extracted from a task delegation graph, represent process execution examples that are abstracted from a specific process instance. Such task patterns can be reused in recurrent cases to reconstruct an ad-hoc process according to the captured previous example flow in the sense of the programming by example paradigm [Cyp93, Lie01]. The task pattern model is shown in Figure 4.6 and discussed in the following sections.

4.5.1 Task Pattern Structure

Parent/sub-task self references of task patterns, enable hierarchical decomposition similarly to task instances. However, in contrast to task instances, task patterns do not require a root task because they do not belong to a concrete process instance. Instead, task patterns represent reusable building blocks for ad-hoc task instances, which describe how a given generic business task can be accomplished. Task patterns can be defined to different level of details in terms of hierarchical decomposition and task context information depending on the users' attitude towards managing reusable task and process data.

Suggested task pattern association in a given task pattern points at a task pattern, which can be used for further guidance and task decomposition, and which may be applicable also for other (similar) business cases. For example, a generic task for organizing a workshop can involve a task for ordering food, including finding a suitable catering company, comparing menus and prizes etc. The task for ordering food can be also relevant for another generic task for organizing a Christmas party. Hence, the food ordering can be stored as a separate task pattern and provided as suggestion for the food ordering tasks in the top-level task patterns for both – workshop and Christmas party organization. Task pattern decomposition and suggestions are discussed in Chapter 5.

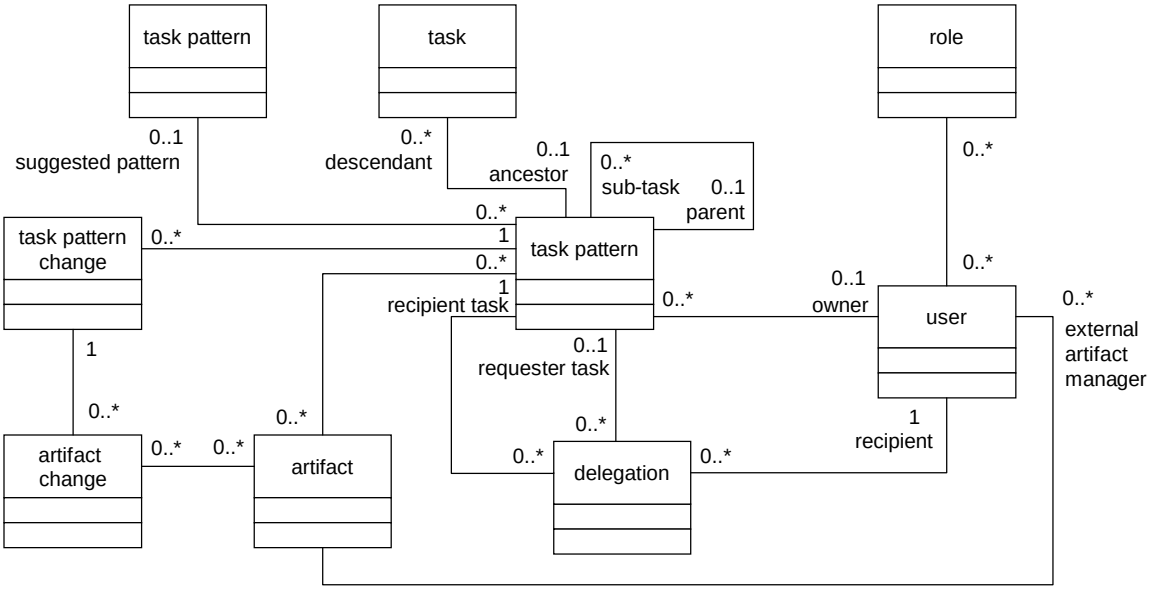


Figure 4.6: Task pattern model

4.5.2 Task Pattern Attributes

Task patterns aim at providing guidance for recurring cases by abstracting from a concrete process instance. Therefore task patterns do not contain attributes, related to concrete task instances. The task pattern attributes are shown in Figure 4.7.

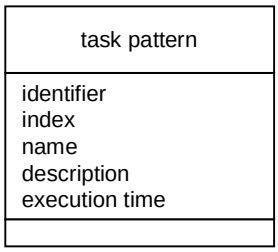


Figure 4.7: Task pattern attributes

The **identifier** servers to uniquely identify a task pattern in the task management system. Identifiers support overall the associations between the task management model entities.

The **index** attribute identifies the index at which a task pattern resides in the sub-task hierarchy of a parent task pattern. This attribute thus supports appropriate ordering of sub-tasks in a parent task within task pattern hierarchies.

The **name** and **description** attributes are self-explanatory and provide human-readable context information in textual form.

The **execution time** stores the time, which is needed for accomplishing a task. While users can set start and due dates for task instances (cf. Figure 4.3), these dates may not match the actual time needed. Dichotomies between target and actual execution times are discussed in related task management models [GOR+07]. Similarly to the latter study, the thesis suggests capturing actual execution time based on the task change history, i.e. calculating time between changes that indicate task processing and task completion.

4.5.3 Task Pattern Changes

Task patterns can undergo various changes. Similarly to task instances, these changes can be structural changes, changes in the context attributes or artifact changes. Artifacts can be associated to task patterns to enable artifacts' reuse in recurring cases. The association of artifacts depends on the type of the declared task pattern and is discussed in the method for composition of weakly-structured process models in Chapter 5.

Tracking of task pattern changes can help to see how a task pattern definition has evolved over time. The thesis considers that a task pattern keeps the up-to-date knowledge of how a specific business case needs to be handled, whereas different task pattern variations are derived for variations in the business case. If a single task pattern is used and updated for different business case variations, eventually the frequent changes can make the task pattern unreliable for any of these variations. Hence, best-practice variations need to be represented through different task patterns rather than through the history of a single task pattern. As a result, task pattern changes are introduced here for completeness. The focus in the thesis is set on managing different best-practice variations that are reflected through different, interrelated task patterns and the management of task pattern change history is not discussed further.

4.5.4 Delegation Flow

As task patterns aim at providing guidance for recurring cases by abstracting from a concrete process instance, they do not involve collaborative delegation flow like task instances. Messages and dialogs related to ad-hoc task instances are considered as depending on the given execution context of the respective ad-hoc process. When a task pattern is extracted from a task delegation graph, the messages and dialogs related to ad-hoc task instances are removed from the pattern. In order to enable reuse of recipient information and recipients' tasks, the task pattern model considers transformation of *task delegation dialogs* to *delegation* entities (cf. Figure 4.6). The transitions between task instances and task patterns and the different ways to compose task patterns are discussed in Chapter 5. Here the focus is set on the underlying model entities.

The *delegation* entities exclude the message flow but inherit a reference to the requester task, the recipient, and the recipient's task which has resulted from the acceptance of the associated task request. Hence, in task patterns recipients are associated to a requester task through task delegation entities. Requester information is not considered in delegation entities because a task pattern provides a task model, where no information is available who will be the actual requester for the delegated task in an ad-hoc task instance that results from this task model.

Thus, in case of task pattern extraction from a task delegation graph, a delegation entity defines how a requester task has been handled by a single recipient. In case of multiple recipients, one requester task is associated to multiple delegation entities. Each of these delegation entities specifies how the delegated requester task has been handled by one recipient through one (accepted) recipient's task. The different interpretations of task delegation towards consolidating process knowledge from multiple recipients are discussed in more detail in Chapter 5.

4.6 Artifacts

Artifacts represent resources, which are used or generated during task execution in the sense of the adopted business process definition (cf. [vdABV+99, vdAvH02] and Definition 1.3). Particularly, artifact associations to task instances and task patterns address the requirements for enhanced document management in the context of task management (cf. Table 2.13, $R_{q\ 2.4}$). The thesis adopts the following definition of an artifact.

Definition 4.3: (Artifact) An artifact as a file such as a text document, a spreadsheet or an executable file, which can be attached to an explicit task representation or collaborative (email) message in a business task management system.

The above definition uses the term "task" to point at explicit task representations of both, a task instance as well as a task pattern. In related task management models artifacts are discussed as attachments that enable resource references in tasks [GOR+07]. The latter study focuses on the roles of attachments for the associated tasks, i.e. whether the respective resources are input or output for the tasks, or if the artifacts are required or related to a task. Specification of such relationships exceeds simple context provision for tasks and goes towards constraints specification (cf. [Ber00]). Setting of constraints requires explicit cognitive efforts and is not achievable through simply attaching documents to to-do items or email messages. Therefore artifact roles as discussed in [GOR+07] are not considered in this thesis. Instead, the focus is set on common artifact operations such as adding, removal and update of artifacts.

Adding, removal and update of artifacts in a task instance or a task pattern are considered as an artifact change. As artifacts are part of the task context information, artifact changes imply also a task change. This is depicted through the relationship between the task change and the artifact change entities in the task instance and task pattern models (cf. Figure 4.2 and 4.6). The following artifact change types are considered: *artifact added*, *artifact removed*, *artifact changed*. In a single task editing operation, more than one artifact changes can be made, i.e. by adding, removing or updating different artifacts. The effects of the artifact changes on a task instance depend on the artifact type. Three basic artifact types are discussed in the thesis, which relate to three different aspects: (i) flexible document management, (ii) knowledge management, and (iii) privacy [SSS07]. The artifact types are discussed in the following.

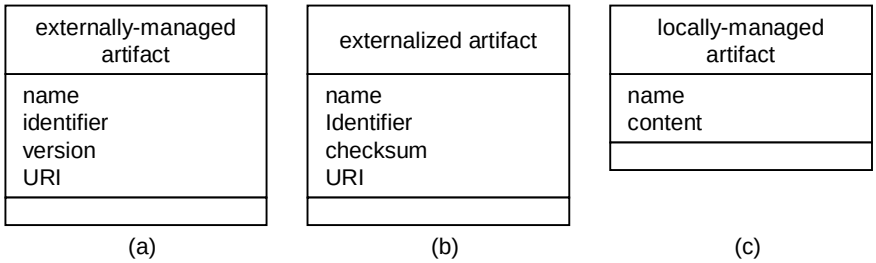


Figure 4.8: Attributes of: (a) externally-managed artifact; (b) externalized artifact; (c) locally-managed (non-externalized) artifact

4.6.1 Externally-Managed Artifact (EMA)

An Externally-Managed Artifact (EMA) is an artifact, the content of which is managed by a user or a user group outside of the scope of a task instance or a task pattern. An EMA can be a document, such as a whitepaper or a technical report, which is being elaborated (authored) by multiple users in the context of a concrete process. The thesis sets the focus on business process

composition based on task management and does not discuss collaborative authoring techniques [HRD+06]. Such authoring is considered as a peripheral activity that affects the resources associated to a given task instance. Another type of EMA can be a document, which is provided as a template from a company department and is used in various processes throughout an enterprise. Such could be e.g. an employment contract template, which is provided by a human resources department.

The user or user group, managing the artifact content, are referred to as *external artifact managers*. An EMA can be associated to one or more external artifact managers (cf. Figure 4.2 and 4.6), who can edit the artifact content in their workspaces and submit a consolidated EMA version to a globally accessible artifact repository. The required repository structure for storing task, artifact and human actor information and the interrelation between the different repositories are discussed in Chapter 7. For the discussion of the task management model here it is important to stress that the underlying server and repository infrastructure enables interrelation between task, artifact and user entities as described in the task instance and task pattern models.

One or more EMAs can be associated to one or more tasks. This association enables among others detection of similar tasks based on the usage of similar resources. Tasks that refer to the same EMA can be inspected for similarity towards document-based proactive information delivery and knowledge management [HRD+06].

The **EMA attributes** are shown in Figure 4.8 (a). An EMA has a human-readable name and a unique identifier throughout the process composition environment. The EMA identifier and version determine the EMA content that is relevant for a given task entity. This content can be retrieved from the artifact repository by using a Unified Resource Identifier (URI), pointing at the actual location of the EMA content within the task management or associated document management system.

An **EMA artifact change** entity inherits all attributes of the EMA shown in Figure 4.8 (a). When an artifact change of type *artifact added* occurs for a task all change attributes are assigned with values so that the complete information about the newly associated artifact is stored.

EMA changes of type *artifact changed* preserve the original identifier attribute but may alter the dynamic attributes name, version or URI, i.e. if the EMA name or content are changed through external artifact managers. During such changes the EMA version on the artifact repository is increased and notifications are triggered to all associated tasks (instances and patterns). An owner of such a task is able to switch the reference to the updated EMA version or preserve the current EMA reference.

EMA changes of type *artifact removed* for a given task occur when an EMA association is removed from the task. An artifact change entity of this type stores all attributes of the removed EMA to uniquely identify the removed artifact and the concrete content (version) of the artifact that was associated with the task. Removal of the EMA association from the task does not remove the EMA from the artifact repository. Users with administrative or managerial rights can perform regularly auditing of this repository and decide whether to remove EMAs that are no longer referenced in any tasks.

4.6.2 Externalized Artifact (EA)

While EMAs enable enhanced flexibility in the document management, they require also cognitive efforts for explicitly editing and submitting artifacts to the artifact repository and for managing those artifacts. In order to ensure unobtrusiveness and in the same time to capture the document flow in ad-hoc processes, the thesis introduces Externalized Artifacts (EAs). An EA is added by an end user as a common attachment to a task instance or a task-related message in the local workspace and externalized (replicated) to a central server infrastructure. Through this the artifact is available in the global process scope beyond the personal workspace. The thesis suggests enabling externalization in an unobtrusive manner, without additional user effort by tracking user actions on task instances and messages in the local workspace and replicating task

attachments to a central artifact repository. As EAs focus above all on unobtrusive capturing of the document flow for end-user driven business process composition, they are also the primary artifact type considered in the dissertation. EAs are relevant also for task patterns. The association of the different types of artifacts in task patterns is discussed in more detail in Chapter 5.

The **EA attributes** are shown in Figure 4.8 (b). During artifact externalization a single artifact copy, identified in a unique manner, is created in the artifact repository for artifacts with the same name and the same content. Thus an EA has a human readable artifact name, a unique identifier throughout the process composition environment (including artifact repository), a checksum for comparing existing EAs in the artifact repository with submitted artifacts, and a URI pointing at the concrete location of the EA content in the artifact repository. The checksum can be determined also dynamically based on the EA content and name. The checksum attribute is included explicitly in the EA attributes' list to emphasize that during externalization equal artifact content and name result in retrieval of existing EA from the artifact repository. A new EA entry is created only if an EA with the given checksum does not exist during externalization.

The comparison of tracked artifacts with existing EAs during externalization and the reuse of the latter enable analysis of similar tasks based on the usage of similar resources analogously to EMAs. Externalization hence can enable unobtrusive detection of recurring tasks and recognition of global optimization possibilities based on usage of similar resources in dispersed, independent processes. A further consequence from task externalization is that in case of extraction of a task pattern, the latter can contain only a reference to EAs in the artifact repository. These references provide a system dependent association of artifacts within reusable task structures. As a consequence, artifacts are not provided outside of the system context and the appropriate artifact access policy. When a task pattern is reused, artifact content can be retrieved from the central artifact repository based on the unique identifier and/or URI.

An **EA artifact change** entity inherits all attributes of the EA that are shown in Figure 4.8 (b). When an artifact change of type *artifact added* occurs, all change attributes are assigned with values so that the complete information about the newly associated artifact is stored.

EA changes of type *artifact changed* are not considered for EAs because such changes would affect the artifact name or content. These attributes determine the checksum of an EA and through this identify the EA itself. Thus, if an artifact, which was previously attached to a task and externalized, is edited by the user in the local file system and attached again, after externalization this artifact is identified as a different EA than the originally attached one. Hence, artifact changes of type *artifact change* for EAs are replaced through *artifact removed* and *artifact added* changes.

EA changes of type *artifact removed* remove the association of the task to an EA but preserve the EA in the artifact repository if the EA is referenced also in other tasks. If no references to the EA exist, the EA is removed from the artifact repository.

4.6.3 Locally-Managed (Non-Externalized) Artifacts

The access policy for artifacts in the artifact repository might not suffice for the privacy needs of end users in different business domains and cross-functional areas. Therefore a possibility to store artifacts in a local, non-externalized manner is considered. As shown in Figure 4.8 (c) such artifacts only have a name and a locally stored content. The latter can be embedded directly into a task (instance or pattern), e.g. as binary content. Tasks using such kind of artifacts however do not benefit from the global data distribution and unobtrusive knowledge management enabled through EAs and the additional flexibility provided through EMAs.

Artifact changes for locally-managed artifacts are limited to the *artifact added* and *artifact removed* change types because these artifacts are decoupled from the repository system and do not allow versioning and tracing of content changes.

4.7 Human Actors

In related literature human actors are considered as resources for tasks in the context of business processes [vdABV+99, vdAvH02, Jor04]. Related task management models [GOR+07] suggest explicit setting of generic, task-centric roles to task items for characterization of specific perspectives towards tasks in a task management system, which require different functionalities and permissions. For example, [GOR+07] consider roles such as *"creator"*, *"owner"*, *"internal observer"*, *"external observer"*, *"controller"*, *"analyst"* that can be used to adjust the behavior of a task management system by providing role-specific support for various operations such as accessing, monitoring, escalating or analyzing task items. The roles proposed in [GOR+07] have been developed jointly with the author of the thesis and [SS08b] reports first steps towards extending the discussed task pattern model with such task roles for modeling the interactive behavior on explicit task representations in a task management system. The latter study describes work in progress which has not been evaluated yet and which is not included in the thesis.

While such task roles can be useful for modeling the interactive behavior of the task management system, evidences show [BDG+04] that in the context of personal task management tasks are recorded in a rather underspecified manner without spending much effort on providing detailed task information. Moreover, the boundary conditions of ad-hoc, unplanned tasks in knowledge-intensive processes are initially ill-defined and change frequently in the course of an ad-hoc process [HMBR05]. The explicit effort for declaring roles and constraints may not pay off over time, which inhibits the end users from modelling their work in detail [RRMvdA05]. Hence, it can be hardly assumed that end users will engage with detailed specification of role-based or other constraints on task items on regular basis. Specification of roles as discussed in [GOR+07, SS08b] resides rather in the constraint provision spectrum [Ber00] which the thesis does not consider relevant for end users. Roles can be used by local developers to set up the task management environment or to model the interactive behavior of the task management system for given processes. On the other hand, unobtrusive task management needs to consider accessible human actor information from the end users' working environment [HMBR05].

4.7.1 Accessible Human Actor Information

The introduced task management model considers simplified, light-weight representation of human actors associated to tasks with the major purpose to store knowledge about the persons, who have expertise related to a given task. This knowledge is important for unstructured, ad-hoc work, where *"employee's key asset is their network of contacts and those people they can approach for advice or help"* [RJS02]. The presented task management model relies on gathering human actor information from existing users' working applications such as email and to-do lists. These applications do not provide domain-specific user roles such as organizational or security roles per se. The thesis considers that common, accessible information is restricted to user email address and name. The respective attributes of the user entity are shown in Figure 4.9.

The *address* attribute is considered unique for a given user and serves as user identifier throughout the task management system. User information is captured from the working applications, and replicated and managed in a central user repository. The underlying repository structure is discussed in Chapter 7.

The presented task management model considers the limitations of users' working applications and avoids introducing domain-specific roles. Such can be added in concrete system implementation where each user can be assigned various organizational and security roles. This is depicted through the association between the *user* and the *role* entities in the runtime task management model (Figure 4.2) and in the task pattern model (Figure 4.6). The overall task management model considers task-centric user roles, as discussed in the following.

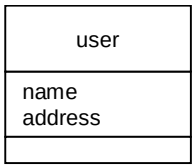


Figure 4.9: Attributes overview of a user entity

4.7.2 User Roles

From task-centric perspective, the task management model considers two basic association types, i.e. two task-centric roles, for human actors – *owner* and *recipient*. These roles are relevant for both, task instances and task patterns (cf. Figure 4.2 and 4.6). For task instances additionally a *requester* role is considered to complement the representation of collaboratively handled tasks in the local user workspaces for both - users who have delegated tasks and users who have received delegated tasks. The meaning of these roles is introduced briefly in the following.

The **owner** of a task in the context of *task instances* is a person, who's to-do list contains the task, i.e. who is responsible for the task execution. In the context of *task patterns*, the owner provides a suggestion about the person who has generic expertise about a given task and in whose responsibility area the task resides. When a task pattern is extracted from a task instance, owner information is inherited. In the resulting task pattern this information identifies the person, who had the task in their to-do list during an actual process execution.

A **recipient** is a person, who has received a task through a delegation from another user. When a user delegates a *task instance* from their to-do list, recipient info entities are generated, which store information about the recipients and their tasks in the local task instance of the requester. In the context of *task patterns* recipients suggest the persons who can process a given task. Transfer of human actor information between task instances and patterns is discussed in Chapter 5.

A **requester** is a person, who has requested a given task in an ad-hoc process instance. This role is relevant only for *task instances*. Requester information is stored in tasks to enable the users to view, from whom they have received a given task, in order to enhance the recipient's view on an accepted task.

4.8 Scientific Achievements

This chapter has introduced a task management model that supports end-user driven business process composition through explicit user interaction with light-weight, simplified task items pertaining to personal task management. This approach allows end users to structure and manage work items according to their actual process knowledge and working strategies. Thereby users are able to establish a direct relationship between their actions and the emerging processes, thus becoming *"informed participants"* in business process composition [FGY+04]. The aspect of keeping end users aware of emerging processes and enabling them to influence these processes at run-time is not supported in process mining approaches [vdAW03, vdAW05] which focus on process discovery rather than on process tailoring by end users. Furthermore, by giving end users full control over the emerging processes, discrepancies between the actual structure and context information of emerging processes and the structure and context information that is expected by end users can be reduced. Such discrepancies accompany attempts to enhance personal task management through contextual support and task (pattern) mining [RC07].

User-centric process support in light-weight personal to-do lists is discussed by a large body of research [Ber00, Jor04, HMBR05, HRD+06]. An additional aspect that is considered by the elaborated task management model is to enable a gentle slope of complexity [MCLM90] for

process tailoring by end users through integrated support in the common users' working applications for task management (to-do lists) and collaboration (email). The task management model enables process composition based only on interrelations between to-do items and email messages. The task hierarchies of different process participants are bound to task delegation graphs based on the email exchange for task delegation. Unlike approaches for modeling of collaborative processes through visual languages [Swe93, GH98, Jor04], task delegation graphs do not require any initial process model or preliminary knowledge of a process. Task delegation graphs are themselves emerging process models which unfold during end users' task management activities and task delegation over email.

Email plays a central role for the composition of weakly-structured process models through the introduced task management model. The task management model considers a *language/action perspective* [Win86, FGHW88] and enables exchange of tasks and deliverables through email messages pertaining to basic speech acts. The model introduces intermediate task states to support the speech acts for exchange of deliverables. Such states are not considered in related studies on ad-hoc process support [GOR+07, Dus04]. Task-related messages are bound to task delegation dialogs which represent larger conversational structures. The actual discourse takes place in the email text, which allows open-ended collaboration and does not submit user behavior to strict speech act rules [But94]. Thus the task management model leverages the role of email as a central organizational interface [GRM+04, BDH+05, SIT06] towards business process composition. This aspect is neglected in related work on user-centric process support [RRMvdA05] where email is considered as a competitive rather than as synergetic environment for process composition.

Collaboration support with a gentle complexity slope for the end users is further provided through different, locally available requester and recipient task instances for a collaboratively handled task. Such different task instances are not considered in related work on user-centric process support, where a single task instance is transferred between different users or accessed in a shared manner [ADMG97, Ber00]. Through locally available requester and recipient task instances users are not required to enter proprietary environments for collaborative task and process management. Instead, a single entry point to the overall process, and a reference to the dependent task of an immediate collaborator are available in the local workspace of each user. The local information about the further processing of dependent tasks provides incentives to end users to enter a detailed process overview. This overview allows users to extend their skills with common task management applications towards ad-hoc process analysis and decision support.

Abstraction of dialogs for exchange of tasks and deliverables into delegation entities in reusable task patterns is defined through the provided task pattern model. This abstraction removes task details that are relevant only in the context of a concrete ad-hoc process instance but preserves the recommendations about general expertise and further task decomposition. Such transformation is not considered in related task management and task pattern models [GOR+07].

Interrelating ad-hoc task instances with derived workflow task models is further enabled through the introduced task management model. These interrelations are established and used during the transformation of weakly structured process models to formal workflows. Such relationships are not considered in related task management models [GOR+07] or studies on integrated support for structured and ad-hoc work [Ber00, Jor04]. The latter studies enable process automation by enforcing constraints on task hierarchies but do not enable transformation of user-defined task hierarchies to structured workflow models.

The spectrum of scientific contributions of the introduced task management model is further extended through the methods for composition of weakly-structured and structured process models discussed in Chapter 5 and Chapter 6. Both methods use the entity relationships and capabilities provided by this task management model. Combined assessments of the scientific achievements are given at the end of the respective method chapters.

4.9 Summary

This chapter has introduced the basic approach for end-user driven business process composition and a task management model for supporting this approach. The approach is based on personal task management and suggests composition of end-to-end business process models by interrelating personal task hierarchies of different end users based on email exchange for task delegation. Enterprise processes unfold in an implicit, unobtrusive manner through user-driven coordination and task decomposition, where each user is modeling the process in their area of expertise by managing an individual to-do list of tasks in the local workspace.

A runtime task management model has been introduced. It describes the underlying concepts and relations for management of to-do items and aggregation of task, artifact and human actor information for the composition of weakly-structured process models. The data is aggregated during the end users' personal task management in the course of evolving ad-hoc processes. The model further enables aggregation of collaborative flow on tasks into task delegation dialogs, which enable transparency into the task-related collaboration. Associations to generic task model and task instance entities of structured workflows have been further discussed.

A task pattern model has been further described, which complements the overall task management model at schema level. Task patterns can be extracted from captured ad-hoc process instances by abstracting from concrete process instance data towards the definition of best-practices for recurring business cases. Task patterns can be further adapted at design time and reused as models for ad-hoc task instances in informal business processes.

Different types of artifacts have been introduced for supporting flexible association and update of resources in tasks, unobtrusive replication of document flow in emerging collaborative processes, and increased privacy for confidential artifacts.

Human actors have been discussed with respect to generic, task management centric user roles. These roles consider existing capabilities of users' task management and email environments and do not rely on domain-specific organizational roles.

The dynamic behavior for composition of weakly-structured process models based on the introduced task management model, as well as the transitions and interrelationships between task instances and task patterns are discussed in detail in Chapter 5.

CHAPTER 5: A Method for Composition of Weakly-Structured Process Models

This chapter presents a method for composition of weakly-structured process models for supporting ad-hoc, human-centric business processes. The method uses the task management model introduced in Chapter 4 to aggregate data from the underlying users' applications for task management (to-do lists) and collaboration (email). This chapter discusses the dynamic behavior of the task management model entities which leads to the composition of task delegation graphs and dialogs. The transitions between task instances and task patterns are further discussed.

5.1 Task Instance Management

In order to compose end-to-end enterprise processes that spread beyond the personal workspace of end users, the thesis considers replication of task, artifact and user information to central repositories. This replication is needed to capture end users' activities on explicit task representation in a task management system on personal as well as organizational level for enabling programming by example [Cyp93, Lie01] of weakly-structured process models, i.e. capturing and repeated execution of ad-hoc business processes. Replication is reported by various related research approaches on user-centric process support [ADMG97, Ber00]. For clarifying the distributed handling of the task management model entities, a client and a server layer are considered in the following. The client layer refers to the applications for personal task management in the end users' workspace, and the server layer refers to the central enterprise software infrastructure where task management data is replicated and stored (cf. Figure 4.1). In this chapter no differentiation is made between middleware server applications and the repository structure. The underlying architecture is discussed in Chapter 7. For the discussion of the process composition method here it is considered that the repositories and the related server functionality enable the associations between the different entities of the task management model as discussed in Chapter 4. In the following the basic task instance operations are presented, which define also the composition of end-to-end business process models based on personal task management.

5.1.1 Task Instance Creation

For ensuring unobtrusive support and a gentle slope of complexity [MCLM90] for process composition by end users, the creation of task items relies on a common functionality from the end users' task management environment. Task instances result from the creation of task items in a light-weight to-do list (R1) in the local workspace of end users. For composing end-to-end business processes in the form of task delegation graphs (cf. Figure 4.1), the task management environment replicates task data to a central server infrastructure. Thus, creation of task instances considers a local (client) and a remote (server) perspective on task instances. These two perspectives may involve complex exception handling and rollback mechanisms to ensure data consistency on the client and the server. Such mechanisms are not discussed in the thesis as they strongly depend on the actual implementation, e.g. whether a synchronous or asynchronous client/server communication for task data replication is chosen. The functional flow for task instance creation is shown in Figure 5.1 and discussed in the following.

Task instances are specified as to-do items in an input form, where the different task instance attributes are set. This form may be provided through the common users' working applications. For example, Microsoft Outlook provides tasks and respective forms that have all human-readable attributes of the discussed task instance model (cf. Section 4.4.1.4). The task form is opened when a new task item is created from scratch or when a user chooses to accept a task

request. During task item creation the process composition environment adds also the meta-attributes: root task, parent task and index. These are used later on to maintain the task structure on the server but also for arranging tasks on the client. After the user fills the task data and saves the task, the task is tracked on the server, i.e. all task's attributes and associations to artifacts and human actors are transferred to the server instance.

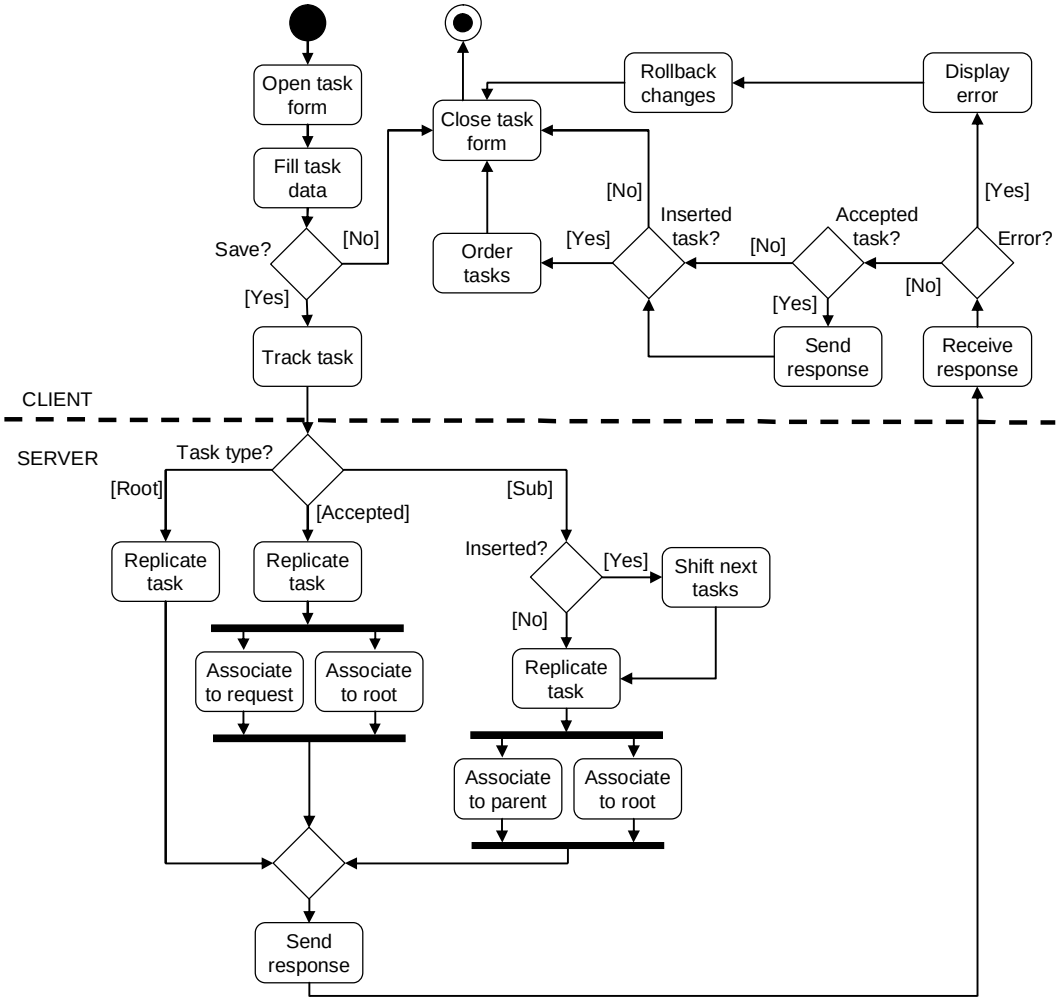


Figure 5.1: Task instance creation

Root task tracking replicates the respective task instance as a starting node for the potentially upcoming task delegation graph (collaborative process). All subsequent sub-tasks and delegated tasks for the process are associated to the created root task instance.

Accepted task tracking associates the accepted task instance of the recipient to the respective request message for task delegation (or negotiation message if negotiation was initiated). Tracking of messages for transfer of tasks and deliverables is discussed later on in more detail. Accepted tasks are further associated to the overall root task in order to construct the task delegation graph, i.e. to associate all collaborative tasks to a single process instance.

A **sub-task** can be generally appended as a last element in the sub-tasks' collection of a parent task, or inserted between the currently existing sub-tasks. In the latter case, the index attributes of the subsequent sub-tasks in the parent task's collection are increased during tracking to ensure a correct order of the sub-tasks. The added or inserted sub-task then needs to be associated to the

parent task, for ensuring correct hierarchical structure, and to the root task for assigning the sub-task to the process instance. The root task relationship can be established also by tracing backwards the parent tasks to one that has a root task association, i.e. the root task itself or an accepted task. However, this can lead to significant performance losses when querying the process instance for a given sub-task in large task hierarchies. Therefore the method suggests establishing directly a root task relationship in all sub-tasks upon their creation.

After a task instance has been processed on the server, a response is sent to the client. This response can denote a failure or success of the tracking operation. In the first case, a human-readable error description is shown to the user, to inform them about possible inconsistencies. A rollback of the changes on client side follows, where users are enabled for example to store a draft of the newly created task so that they do not lose the information.

If the tracking of the task was successful, the server response remains transparent for the users in order to ensure unobtrusive tracking. Creation of an accepted task triggers a response message, which indicates explicitly the acceptance and is sent back to the requester.

Independently from the type of the created task (root, accepted or sub-task) the to-do list application may need to order (arrange) the existing tasks. This depends on the provided to-do list functionality. For example Microsoft Outlook and other task list managers [BDG+04] do not provide hierarchical task ordering. Integration of the process composition environment in such tools would require task ordering based on structural criteria. These criteria may be provided through the root, parent task and index attributes of the introduced task management model.

5.1.2 Task Instance Editing

Task instance editing depends on the concrete office integration environment. If task management is available in the users' working applications, the process composition environment should use it to provide integrated process support with a gentle complexity slope [MCLM90].

Task instance attributes are generally edited in task forms which are provided by the concrete office integration environment. All task instance changes are tracked on the server to update the task delegation graph.

Artifacts editing is different for the different artifact types. *Externally-managed artifacts* require enhanced functionality for creation, retrieval and publishing of globally accessible files. For this purpose, a document management system needs to be used [Jor04, HMBR05]. Adding of such artifacts is related to a perceptible cognitive effort, as users need to have knowledge of the concrete document management system and to act in it consistently by performing document management along with their task management activities.

Externalized artifacts on the other hand are added as common attachments with reduced cognitive effort. The thesis focuses on this artifact type, as it is especially relevant for providing unobtrusive process composition and capturing document flow without additional user efforts.

Locally-managed, non-externalized artifacts are considered in the thesis as a special case with focus on privacy. The user is allowed to explicitly state, which of the artifacts they wish to preserve as locally-managed artifacts. In this case, the artifacts are dissociated from the task instance on the server if these were externalized, and stored locally with their full content.

5.1.3 Transfer of Tasks and Deliverables

For transfer of tasks and deliverables the thesis suggests the usage of computational mail [Bor92]. Computational mail enables unobtrusive process support by embedding task data in the common email infrastructure. In related literature [ADMG97] computational mail is used for email-based workflow support through active software objects such as mail-robots, agents and applets. In contrast to [ADMG97] the thesis suggests embedding of task-specific meta-information rather than of active software objects. The embedded meta-information is interpreted by the process composition environment to update the emerging process model in a centralized manner. The

embedded message attributes and the functional flow for message processing from system's perspective are provided in Appendix C.

The transfer of tasks and deliverables is performed through exchange of email messages. The collaborative task handling from requester's and recipient's perspectives is shown in Figure 5.2.

Create steps denoted with a light-gray background and bold titles involve pre-processing of task instance and message objects. Pre-processing is performed on task-related emails independently of the concrete message type to enrich the messages with task-specific information. For example, task requests inherit attributes and context information from the requester task (cf. Section C.2). Responses to previous messages, such as negotiations, acceptance and declination of requests, inherit attributes from the messages to which they respond. On the other hand, when an accepted task is created, the task instance is pre-processed to transfer request attributes. The different message types are discussed in more details in the next sections.

Send steps denoted with white background and bold titles in Figure 5.2 involve post-processing of messages, which is performed on all task-related emails to track them on the central server instance (cf. Figure C-1). Post-processing of emails sent by a task requester additionally updates the recipient statuses at requester site as discussed in the next sections.

Receive steps denoted with a dark-gray background involve system actions for updating task (recipient) statuses at requester site, i.e. when a message from a task recipient is received.

Inspect steps denoted with normal background and titles involve user interactions for inspecting human-readable task and message information.

Execute steps involve abstract activities that can be performed outside of the task management system.

The different steps in the collaborative task handling are discussed in the following by referring to the requester and recipient perspectives (Figure 5.2).

5.1.3.1 Request

Requests are issued for existing task instances from a requesters' to-do list, i.e. before delegating a task the user needs to create a task entry in their to-do list. This is required in order to have a local task representation of the collaborative task in the workspace of the requester. Through this the requester is able to keep track of all tasks that they have delegated to other persons, without the need to leave the personal workspace. This consideration aims at ensuring a gentle slope of complexity [MCLM90] for process tailoring by end users.

When a request is triggered, the request message is pre-processed to transfer data from the task instance. This data includes meta-information with system attributes for associating the task and message entities according to the runtime task management model, as well as human-readable task attributes such as subject, description, start and due dates, priority etc. (cf. Section 4.4.1.4) of the task request. The visual presentation of the human-readable task attributes depends on the provided email functionality, i.e. whether only plain text is provided or enhanced formatting e.g. through HTML email bodies. The user is enabled to edit the request as a common email message.

Tracking of a request message creates a dialog for each of the specified recipients, replicates the request message and associates it to the requester task on the server (Figure C-1). After a successful request tracking, the recipients' statuses of the requested task are set to *requested* for each recipient of the task request. Thus, the requester is able to see which tasks are requested from which users through the *recipient info* entities in the personal workspace (cf. Figure 4.2).

When a recipient *inspects* a task *request* (cf. Figure 5.2), they are able to *negotiate*, *decline* or *accept* the task request. These actions can be characterized as basic speech acts in a *"conversation for action"* according to the language-action perspective [Win86] on which the collaborative message handling is based. The responses are enabled through context-specific controls, which are activated based on the meta-information embedded in the request.

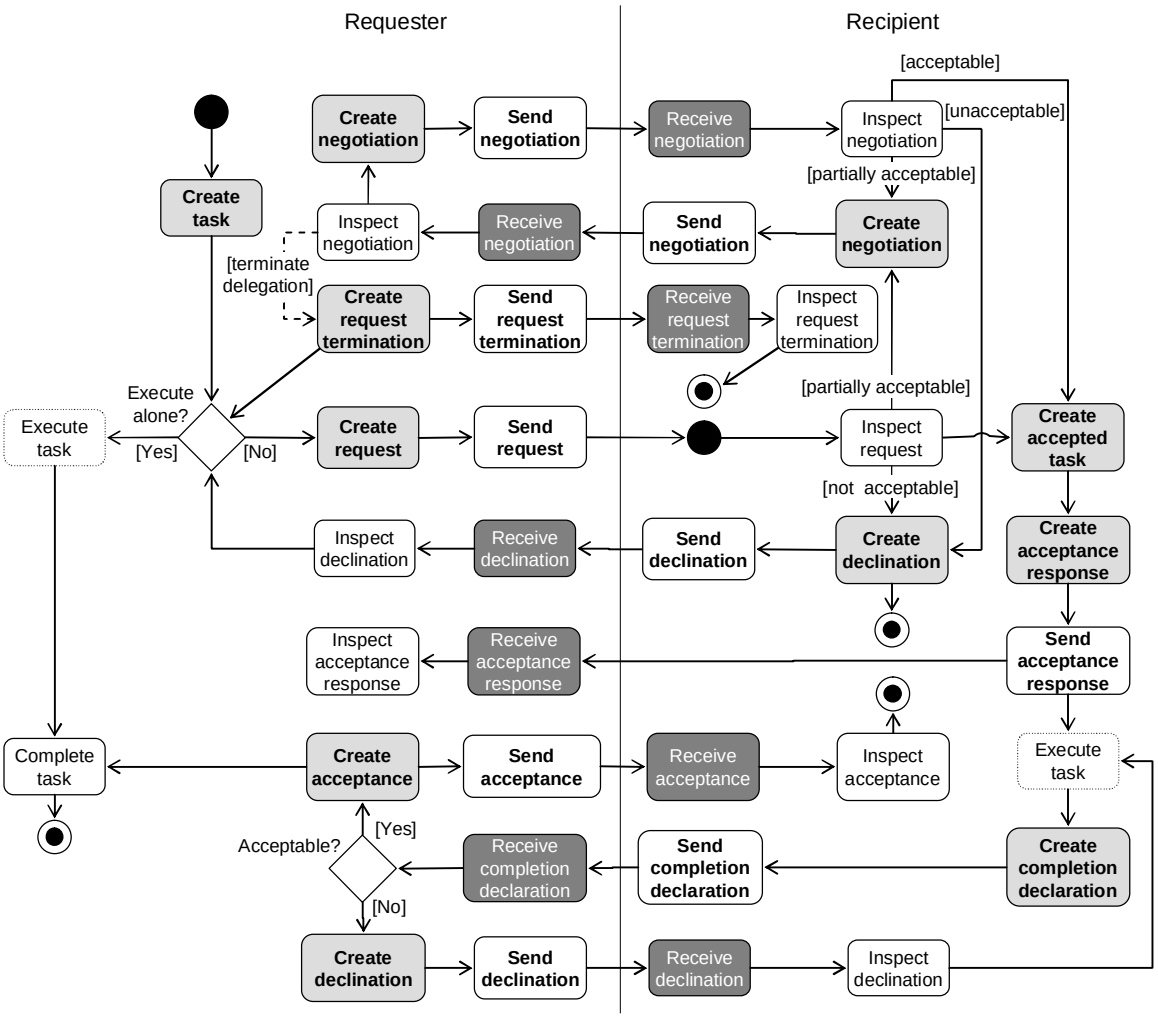


Figure 5.2: Collaborative task handling

5.1.3.2 Negotiation

Negotiation is created for a task request if further clarifications of the assignment are needed. When a negotiation message is created, it is pre-processed to embed the request attributes. When the message is sent, it is tracked on the server. Tracking of a negotiation message replicates the message on the server and binds it to the corresponding dialog and previous message (cf. Figure C-1). The tracking procedure is equivalent for all message types other than requests.

The task requester *receives the negotiation message* (Figure 5.2) with embedded, task-specific attributes that are inherited from the initial request message. These attributes allow identification of the original requester task so that its recipient status can be updated to *negotiated*. The task requester is able to *inspect* the negotiation message and to respond with another negotiation message. The latter may continue the discussion on the boundary conditions of the task assignment. In this case, the recipient may respond with a further negotiation, decline or accept the negotiation. This process continues until an agreement is reached, or the request is terminated or declined as discussed later on.

5.1.3.3 Request Acceptance

Request acceptance is accomplished in several steps. First, a recipient task instance is created, by transferring message data from the request, including attached artifacts and task meta-information to the resulting recipient task. The recipient task emerges as a replica of the requester task, including additional information from the request message. The recipient can edit the accepted task instance as any other task in the to-do list.

Second, a request acceptance message is created which the recipient can edit and send to the requester. This message complements the conversation by explicitly informing a task requester that their request has been accepted. This approach follows the language/action perspective [Win98] which considers a conversation as a *"coordinated sequence of acts that can be interpreted as having a linguistic meaning"*, i.e. through the request acceptance message the message exchange and the conversation are not interrupted.

The explicit acceptance further enables update of the recipient status on requester site. During creation of the request acceptance message, pre-processing is performed to transfer task-specific information from the request message. When the request acceptance response is received by the requester (cf. Figure 5.2), the system parses the embedded task-related information and updates the recipient's status in the recipient info of the corresponding requester task to *accepted*.

5.1.3.4 Request Declination

Request declination messages are pre-processed to transfer the request data to the declination message. When the declination is received at requester site, the system parses the contained task meta-information and updates the recipient status to *request declined*. The requester is able to inspect the message in order to obtain a clarification about the reason for the declination. The requester can then create and send a new task request to the same or to a different person.

5.1.3.5 Request Termination

A requester may trigger a request termination as a response to a negotiation message that is received from a task recipient. In this case the request termination message is pre-processed to embed task-related data and negotiation details from the negotiation message. Alternatively, a requester can trigger a request termination message on a given delegated task for recipients who have not yet accepted the request for that task. As in this case the request termination is not created as response to another message, the transition from the *inspect negotiation* action is depicted through a chain-dotted arrow in Figure 5.2. The request termination message in this case is again pre-processed to embed task details. When the request termination is sent by the requester, the recipient status in the requester task for the corresponding recipient(s) is updated to *request terminated*.

When the recipient receives the request termination message, the embedded task information allows finding the appropriate request or previous negotiation messages at recipient site as request, negotiation, and request termination messages have the same *parent task* association pointing to the requester task (cf. Figure 4.2). The collaborative handling of that task on recipient site is considered as terminated.

5.1.3.6 Completion Declaration

Completion declaration enables negotiation of deliverables. This message is sent back to the task requester when a recipient marks an accepted task as completed. When a completion declaration is sent, the recipient task is set in state *completeness declared* (cf. Figure 4.4). The completion declaration message is pre-processed to embed task-specific meta-information referring to the originating requester task and to the recipient task that has been declared as complete. When the requester receives the message, the system parses the requester task information in order to find

the corresponding task in the requester's to-do list and to update the recipient status for the respective recipient to *completeness declared*.

5.1.3.7 Declination of Completion Declaration

Declination of completion declaration is sent by the requester, if the result declared in a completion declaration is not satisfactory. The declination message is pre-processed to embed information about the recipient task from the completion declaration. During post-processing of the declination message at requester side, the corresponding recipient status is set to *completeness declined*. When the recipient receives the declination, the system parses the embedded recipient task information to find the corresponding task in the recipient's to-do list and to update the task status to *completeness declined*. Through this the recipient can see the acceptance of which task has been declined by their requester. The recipient can further act on the task and declare it again as completed. This process continues until an agreement is met and the requester accepts the completion declaration or the task is cancelled.

5.1.3.8 Acceptance of Completion Declaration

Acceptance of completion declaration is sent by a requester, if the result declared in a completion declaration is satisfactory. The acceptance message is pre-processed to embed information about the recipient task from the completion declaration. During post-processing of the acceptance message the recipient status in the requester's task is updated to *completeness accepted*. When the recipient receives the acceptance message, the system uses the embedded recipient task information to automatically detect the respective task in the recipient's to-do list and to update its status to *completed*. An appropriate human-readable notification is displayed to the recipient to inform them about the acceptance of the completion declaration.

5.1.4 Local and Global Scopes in Task Delegation Graphs

Task delegation graphs bind the personal task hierarchies of multiple process participants to end-to-end, weakly-structured process models. The resulting processes exist in an integrated manner on the central server infrastructure but also in a distributed manner throughout the personal workspaces. This raises issues about effects of local changes on overall enterprise processes, and about the awareness of end users about changes in the overall processes that affect their tasks.

The thesis addresses such issues through considering a *local* and a *global* scope in task delegation graphs. Local and global scopes are considered important for group processes [ADML03] and refer respectively to tasks of a single user and to group tasks. In the thesis local and global scopes have different meaning for task instances and task patterns. The scopes for task instances are defined as follows:

Let $G\ (V, E)$ be a task delegation graph with $E_d \subseteq E$ being the set of edges representing delegations (see Definition 4.1, Section 4.3.2). Let $G_A\ (V_A, E_A)$ be a sub-tree of the task delegation graph G with a root task $A \in V$.

- **Local scope for task delegation graphs** refers to task instances, which are not accepted or delegated, and which do not have delegated sub-tasks. Formally, task A is considered to reside in the local scope if $\neg\exists\, e \in E_d\ (A \in e)\ \wedge E_A \cap E_d = \phi$.
- **Global scope for task delegation graphs** refers to collaboratively handled tasks and comprises task hierarchies of more than one user. Formally, task A is considered to reside in the global scope if $\exists\, e \in E_d\ (A \in e)\ \vee E_A \cap E_d \neq \phi$.

5.1.5 Notifications in Task Delegation Graphs

[Bar97] discusses agile processes as *"situated planning"* and emphasizes that planning does not oppose to working *"in situ"*. [Bar97] further considers plans as *"chains of anticipated goals"*

where "*breakdown situations are not exceptions from work activities but are a natural and very important part of any activity which forms the basis for learning and thus for developing and enhancing plans for future action*". Thus a need arises to support awareness, and situated adaptation of user-defined process models according to the current process context.

For supporting such awareness, the thesis proposes a basic notifications framework which considers local and global scopes in task delegation graphs. The notifications framework provides awareness features for supporting task-related collaboration and procedures for handling operations that have global effects to tasks of multiple participants in task delegation graphs.

5.1.5.1 Task Progress Awareness

Task progress awareness is especially relevant for collaborative activities, where users may need to synchronize on dependent tasks. For ensuring a gentle slope of complexity [MCLM90] the thesis suggests awareness in the local user workspace so that users are not required to enter proprietary environments for process monitoring. Local awareness is provided through the *requester info* and *recipient info* entities of the task management model (cf. Section 4.4.3.3). The progress information of collaborative tasks is retrieved from the server upon tracking and propagated to the task instances in the users' local workspaces. For enabling progress awareness in a requester task, various task-instance based notifications are considered for recipients tasks, such as e.g. *progress change* and *status change* notifications. These notifications are triggered when a recipient changes the respective attributes of an accepted task instance. Status is further changed, when a task is delegated or transferred. The notifications are propagated to the requester's task instance and update its recipient info. Thus a requester is able to see if a recipient has delegated a task further or initiated task transfer. For viewing the complete content, structure and further delegations of recipients' tasks, the requester needs to open the process overview.

5.1.5.2 Task Cancellation and Task Completion

Cancellation and completion of tasks is performed through manual user actions in the local workspace and may affect multiple tasks in the local task hierarchy or in the overall task delegation graph. Therefore local and global scopes are considered for these operations. Cancellation and completion are handled in a similar manner as discussed in the following. To clarify the procedures a task delegation graph $G\ (V, E)$ is considered, with $E_d \subseteq E$ being the set of edges representing delegations (see Definition 4.1, Section 4.3.2), and a sub-tree of the task delegation graph $G_A\ (V_A, E_A)$ with a root task $A \in V$.

Cancellation/Completion in the local scope refers to cancellation/completion of task items in a local do-do list, which are not accepted or delegated, and which do not have delegated sub-tasks (cf. Section 5.1.4). Cancellation/completion of a parent task cancels/completes all sub-tasks iteratively, i.e. cancellation/completion of a task A cancels/completes all tasks in G_A. If a sub-task is in state *completed*, before the *cancellation* of a parent task is performed, the system notifies the user that results for the completed sub-task tasks may have already been delivered. Thus the user is enabled to see if some compensation and rollback of deliveries may be required and to chose, whether to continue with the cancellation.

Cancellation/Completion in the global scope refers to cancellation/completion of collaboratively handled tasks and affects task hierarchies of more than one user (cf. Section 5.1.4). Two basic types of cancellation/completion are considered.

Cancellation/completion by a task requester refers to the cancellation/completion of a requester task or of a parent task of a requester task. When a task A that resides in the global scope (cf. Section 5.1.4) is cancelled/completed, the process composition environment evaluates if there are *recipients'* tasks $B_i \in (E_d \cap E_A)$ that are not cancelled or completed. These tasks are considered as *affected* by the global operation. In case of cancellation, completed recipients' tasks are also considered as affected. Notification for completed tasks on cancellation is needed to

allow rollback of deliverables. The information about recipients' tasks is retrieved through the server which is aware of the current state of all tasks in a task delegation graph.

When *A* is cancelled/completed and affected tasks are detected, a *cancellation/completion request* email message is created. This message embeds the task attributes of the requester task that is being cancelled/completed and allows the user to specify the reason for the requested operation. The message is sent to all affected recipients' tasks B_i, i.e. the notification is task-instance based. The message handling is performed according to the language/action perspective [Win86, FGHW88] as discussed in the task management model in Section 4.4.3.1.

When a *cancellation/completion request* is received for a given affected task B_i this task is set in state *cancellation/completion pending*. The recipient of the message is then able to respond with *cancellation/completion approved, negotiated* or *declined* message. *Negotiations* allow further clarification on the requested operation if it is questionable from a stakeholder's point of view, e.g. if critical transactions that require complex compensation handling have been already triggered. *Declination* applies if the requested operation is considered as completely inappropriate by an addressed stakeholder.

The user that has requested the operation can further issue a *cancellation/completion request termination* message to all affected recipient's tasks, if some discrepancies have been detected (e.g. based on declinations and negotiations), which do not allow for completing the overall operation. Request termination removes the *pending* state and recovers the last state which an affected task had before the *cancellation/completion request*.

Tasks for which *cancellation/completion approved* messages are sent, receive a state *cancellation/completion approved*. All tasks in G_A are set to a final state *cancelled/completed* only after *cancellation/completion* has been *approved* for all affected tasks B_i. Messages for task cancellation/completion are associated to a dialog on the server as discussed in Section 4.4.3.2.

Cancellation/completion by a task recipient is performed, when a recipient has accepted a task and then decides to cancel/complete it. *Completion* issues a completion declaration to the requester as discussed in Section 5.1.3.6. *Cancellation* of a recipient task issues a *cancellation request message* to the task requester. In this message the task recipient can specify the reason for the cancellation. The task requester can then respond with a *cancellation approved, negotiated* or *declined* message to the cancellation request. The recipient can further issue a *cancellation request termination* message to the requester to notify them that the cancellation request is irrelevant and continue work on the accepted task.

If the accepted task that is cancelled/completed by a recipient is delegated further or has delegated sub-tasks, the procedure for cancellation/completion by a task requester discussed above is triggered. However, in this case the *cancellation/completion pending* state is resolved only after approval is obtained from the requester of the accepted task, as well as from all recipients of affected tasks in the global scope.

Reopening of cancelled/completed tasks is considered through the *cancellation/completion pending* states. These require explicit clarification of the cancellation/completion where all stakeholders are expected to meet an agreement whether the task can be finally cancelled/completed or not. The final cancelled/completed state is not considered subject to change. If the cancelled/completed task needs to be performed again, it can be reused as task pattern to create and execute a new instance.

5.1.5.3 Suspending and Resuming Tasks

Tasks are suspended through manually setting their status attribute to *suspended* (cf. Section 4.4.1.5). Resuming is performed by setting suspended tasks to the *running* state, again through manually adjusting the status attribute. Suspending/resuming may affect the tasks of different process participants in a task delegation graph. Thus local and global scopes are considered for these operations in an analogous manner as for cancellation and completion. To clarify the

procedures a task delegation graph $G\ (V, E)$ is considered, with $E_d \subseteq E$ being the set of edges representing delegations (see Definition 4.1, Section 4.3.2), and a sub-tree of the task delegation graph $G_A\ (V_A, E_A)$ with a root task $A \in V$.

Suspension/resumption in the local scope refers to suspending/resuming tasks in a local do-do list, which are not accepted or delegated, and which do not have delegated sub-tasks (cf. Section 5.1.4). In that case, setting a parent task to state *suspended*, suspends all *running* sub-tasks of that parent task. This state propagation runs iteratively over the task hierarchy, i.e. setting task A to state suspended suspends all running tasks in G_A. On the other hand, setting a parent task to state *running* applies this state to all *suspended* sub-tasks of that parent task. During the latter operation the user may be prompted to select which of the *suspended* sub-tasks of the resumed parent task to transfer to the *running* state to allow flexible, partial resumption of sub-tasks.

If a parent task is in state *suspended*, setting any of its sub-tasks to a state *running* applies this state also to the parent task. This state propagation runs iteratively over the task hierarchy. Hence if a task A is in state suspended, setting any sub-task in G_A in state *running* sets also A in a *running* state. Allowing the user to select which sub-tasks of a resumed parent task they want to resume prevents from automatically resuming all *suspended* tasks in G_A when a single task in G_A is started or resumed. Such selection increases the user control in the resumption procedure.

Suspension/resumption in the global scope refers to suspending/resuming collaboratively handled tasks and affects task hierarchies of more than one user (cf. Section 5.1.4). Suspension *affects* tasks with the following states: *running*, *delegated*, *completeness declared*, and *completeness declined*. The current state of a task at suspension is stored for recovery upon resumption. Resumption affects only tasks with the *suspended* state and restores the last state in which a task was residing before the suspension.

Suspension/resumption by a task requester refers to the suspension/resumption of a requester task or of a parent task of a requester task. Such suspension/resumption is handled analogously to the cancellation/completion of a requester task. When a task A that resides in the global scope (cf. Section 5.1.4) is suspended/resumed, the process composition environment evaluates if there are *affected recipients'* tasks $B_i \in (E_d \cap E_A)$ with corresponding affected states (as discussed above).

Suspension/resumption request messages are issued to the affected tasks and the latter are set in a *suspension/resumption pending* state. Recipients of the operation request are then able to respond with *suspension/resumption approved*, *negotiated* or *declined* messages.

The user who has requested the operation can further issue a *suspension/resumption request termination* message to all affected tasks, if some discrepancies have been detected (e.g. based on declinations and negotiations) which make the operation infeasible.

Tasks for which *suspension/resumption approval* messages are sent receive a *suspension/ resumption approved* state. All tasks in G_A are set in a final *suspended* state (upon suspension) or in their *original state before suspension* (upon resumption) only after the *suspension/resumption* has been *approved* for all affected tasks B_i. This handling considers that the person that has delegated a task triggers and manages the overall processing of this task up to the delivery of the final result. Intermediate *pending* states and suspension/resumption request messages allow consolidation of all involved stakeholders on the global operation. For example, if for completing their task an owner of an affected task requires resources that are available only in a limited time frame, suspension would not be acceptable from their point of view. Thus, the requester of the global operation can consider the overall task that they want to suspend as time critical and terminate the suspension. On the other hand, if the owner of an affected task in a resume operation is asked to resume that task, but they have already scheduled other time critical activities and are unable to start working on the task, they can at least state this discrepancy by negotiating the resume request. Eventually, they can approve the resume operation and suspend their task later on, after the global resume operation has finished.

Suspension/resumption by a task recipient can be handled analogously to cancellation/ completion by a task recipient by sending suspension/resumption request messages to the task

requester and maintaining intermediate suspension/resumption pending state for the recipient's task. However, the thesis considers that suspension/resumption of the recipient's task may not affect directly the requester task as the requester is the one who manages the overall task in a top-down manner. Thus suspension/resumption messages for a recipient's task are propagated as notifications and reflected in the *recipient info* in the requester task without requiring explicit consolidation. The requester can review the notifications and decide whether to suspend/resume the overall delegated task or to ask the recipient to resume/suspend their task.

An additional point for resumption of a recipient task is that it is resumed also when any of its sub-tasks is set to the *running* state. If the recipient task is delegated further, i.e. if it is a requester task in a further delegation, the procedure for suspension/resumption by a task requester applies.

5.1.6 Structural Changes in Task Delegation Graphs

Structural changes in task delegation graphs consider a local and global scope similarly to cancellation and completion as they can affect the overall task delegation graph. Structural changes involve inserting, deletion and moving of tasks and task hierarchies within a collaborative process.

Insertion of tasks and task hierarchies is possible into all task instances except cancelled and completed tasks.

Moving tasks has different implications in the local and global scopes. *Moving tasks in the local scope* is performed to rearrange task items in a personal task hierarchy. *Moving tasks in the global scope* affects the tasks of different users and is restricted to preserve the agreements that have been made during task delegation. Concretely, before accepting a delegated task a recipient is able to view the overall process (task delegation graph) and to estimate the context for the acceptance. Moving the initially delegated task to another process or parent task can change the task context and invalidate the agreement, by exposing the recipient to possible escalations, i.e. if the delegated task and therewith also the accepted task are moved into a critical, overdue parent task. On the other hand, if a recipient moves an accepted task as a sub-task into another task, this would extend the process context and may invalidate the agreement for the delegation from requesters' point of view. Therefore the thesis suggests completely restricting the moving of tasks from the global scope. An alternative solution is to extend the notifications framework to support such move operations through consolidations analogously to the cancellation and completion operations in the global scope. Such consolidation then requires *move pending* and *move approved* states, where the move is realized after all affected tasks are set in approved state.

Deleting tasks also considers local and global scopes of collaborative processes. *Deletion of tasks in the local scope* helps the user to organize personal task hierarchies by completely removing unnecessary task items. *Deletion of tasks in the global scope* on the other hand affects the overall task delegation graph. Hence, consolidation mechanisms are needed. The thesis suggests that deletion in the global scope is preceded by cancellation, i.e. tasks in this scope can be deleted only after these are set in state cancelled. Through this a consolidation between all involved stakeholders takes place to ensure the consistency of the overall process. Independently of the deletion scope, the introduced runtime task management model, especially the task instance change entity, considers decoupling task information from the task management system and transferring it into task instance change entities. This allows recovery of task instances for analytical purposes, even if the tasks are removed from the process composition environment.

5.2 Task Pattern Management

For enabling exchange, adaptation and reuse of process knowledge (R4) as well as tracing of relationships between best-practices and running processes (R5) and tracing best-practice

evolution (R6) the thesis suggests the use of task patterns [RRMvdA05, GOR+07]. Task patterns serve as best-practices, captured in the form of hierarchical task structures, and contain all task-related information (cf. Definition 4.2). Task patterns can be extracted from task delegation graphs resulting in process examples that are abstracted from a given ad-hoc process instance. These patterns enable programming by example [Cyp93, Lie01] of weakly-structured process models in that they can be used to reconstruct a task delegation graph by following suggested task decomposition and delegation flow as well as using relevant documents and involving appropriate stakeholders based on the previously captured user activities on explicit task representations in a task management system. Extraction, adaptation and reuse of task patterns enables seeding, evolutionary-growth and reseeding (SER) [FGY+04] of weakly-structured process models, which initially emerge through user-driven collaboration in the form of task delegation graphs.

5.2.1 Local and Global Scopes in Task Patterns

In order to enable SER on personal and enterprise level, two scopes of task patterns are considered – *local* and *global*.

Local scope refers to task patterns which are available in a local, non-distributed manner within the personal workspace of an end user. Thus, a task pattern that resides in the local scope enables reuse of individual task and process knowledge by a single user. In the following, such task pattern is also called a *local task pattern.*

Global scope refers to task patterns which are accessible in a shared manner through the enterprise infrastructure. A task pattern that resides in the global scope can be accessed and reused by multiple users in the enterprise. In the following, such task pattern is also referred to as a *global task pattern.*

The global scope is accessible from the local scope, but the local scope is not accessible from the global scope. Further, globally managed artifacts, i.e. externally-managed and externalized artifacts, are accessible in the local and in the global task pattern scope. Locally-managed artifacts are accessible in the local task pattern scope, but inaccessible in the global task pattern scope.

Local and global task patterns are stored respectively in local and remote task pattern repositories, which are discussed in Chapter 7. The task pattern extraction, adaptation and reuse are discussed in the following sections by taking into account the introduced task pattern scopes.

5.2.2 Task Pattern Extraction

The need for global expertise sharing and knowledge management strategies in organizations is largely perceived [Wii04]. Knowledge management is a critical aspect for business process management, where one of the major problems is related to process discovery [Ver04]. [Bar97] discusses agile processes as *"situated planning"* and emphasizes that a central point in providing situated support for user-defined plans is to *"recognise the function of plans as ways of anticipating and pre-handling events in (working) life based on their recurrent nature, and be able to save and later reuse the experience obtained in handling these events"*. Thus, enterprise processes can be seen as plans, which are improved and extended in the actual execution context. Thereby reuse is a central aspect that needs to be considered to support organizational learning and iterative process discovery and refinement.

The thesis considers extraction of captured real-life processes for further reuse in recurrent cases in the form of task patterns. A task pattern can be extracted from an arbitrary item in a local user's to-do list. The user can extract only the local hierarchy of a selected task instance as a personal best-practice, i.e. as a captured task execution example from the personal workspace that can be used to execute similar tasks. Further, the overall sub-tree (i.e. sub-task delegation graph, cf. Section 4.3.2) for the selected task can be retrieved from the server. Such extraction allows reuse of overall process structures, which have been developed collaboratively by multiple process participants on enterprise level. In the latter case, different task patterns are created for

the tasks of different users. The decomposition of task delegation graphs during extraction of task patterns and the resulting task pattern relationships are discussed later on. In the next section the focus is first set on the transferred information between task instances and task patterns.

5.2.2.1 Transfer of Task Context Information

Transferred task attributes include only the *name* and *description*. All other attributes like *priority, start date, due date, status* and *percent complete* are considered as related to a concrete process instance and the accompanying execution context and are not transferred to task patterns.

Artifacts are transferred to task patterns to preserve all task-related resources for future reuse. *Externally-managed* and *externalized artifacts* are associated to the extracted task pattern independently of its scope. The artifacts can be retrieved from the artifact repository during task pattern viewing, editing or reuse, i.e. according to the repository access policy.

Locally-managed, non-externalized artifacts can be handled in different ways, depending on the declared task pattern scope – local or global. Local task patterns inherit the complete artifact content (in binary form). However, when exchanging local task patterns, the users may not be aware of the confidentiality restrictions for embedded artifacts. Therefore, it is reasonable to consider transforming all artifacts as either externalized or externally-managed. When a global task pattern is extracted from task instances that contain locally-managed artifacts, the latter are either excluded from the task pattern or submitted as externalized or externally-managed artifacts.

5.2.2.2 Transfer of Human Actor Information

The transfer of human actor information focuses on expertise recommendation based on the owner and recipient information of task instances.

Owner information of task instances is transferred to extracted task patterns, based on the user name and (email) address as defined in the task management model in Chapter 4. In task instances the owner is mandatory in order to assign tasks to a given user workspace in task delegation graphs. On the other hand, in task patterns the owner recommends general expertise and does not specify explicitly that only this person should handle the respective task. The task owner specified in a task pattern can be asked for assistance in future cases.

Recipient information is embedded in delegation entities as discussed in the task pattern model in the Chapter 4. These are composed differently depending on whether only a local task hierarchy from a users' to-do list, or a complete task delegation graph is extracted. In the first case, delegation entities embed only recipient(s) information and denote which persons have processed collaborative tasks in a process instance. No further details about how they have performed the requested tasks are provided. If a task delegation graph is extracted, recipient information involves also accepted recipients' tasks, resulting from delegation. The recipient information thus specifies not only who has received a delegated task, but how they have processed it further. The thesis suggests decomposition of task delegation graphs into separate task patterns for each recipient, denoting generic business tasks in different expertise areas. Task delegation graph decomposition is discussed in more details in the following section.

5.2.2.3 Decomposition of Task Delegation Graphs into Task Patterns

The decomposition of a task delegation graphs into task patterns during task pattern extraction is shown in Figure 5.3. Task patterns (*A'*, *B'*, *C'*, *D'*, *E'*) are denoted as derivations of the respective task instances (*A*, *B*, *C*, *D*, *E*). The following important aspects are shown:

- Separate **task patterns** are extracted for the root task *A*, and for all accepted tasks *B*, *C*, *D*, and *E*.
- The **owner** for all tasks in a task pattern is the user, who had the original task instances in their to-do list.
- A **delegation entity** is derived from each task delegation dialog by removing the message

flow and preserving the information about requester task, recipient, and recipient task.

- A **suggested task pattern** is set, which points at the task that can be used to further decompose and handle the delegated task in future process executions.

The provided decomposition of task delegation graphs into task patterns aims to provide incentives to end users to explicitly elaborate on the captured task patterns and to refine them in a meaningful manner. This is discussed in details in the following sections.

5.2.2.3.1 Delegation of One Specific Task to Multiple Recipients

A delegation of one task to multiple recipients can mean that the recipients have to perform the same logical task and deliver a result individually. For example, a chief executive officer requests a quarterly report from several department managers, who have to prepare the reports individually, by following the same general procedure. In this case it is reasonable to have a single guideline for all recipients. By default, this guideline is based on the task decomposition of the first department manager, i.e. in Figure 5.3 task *A_1'* receives *B'* as suggested task pattern.

Based on the delegation entities and associated recipient's tasks of the other department managers, e.g. *C'*, the person, performing the task pattern extraction (the chief executive officer), is able to follow and evaluate all possible executions of the delegated task. This evaluation allows the user to select the most appropriate task for further decomposition and execution. Alternatively, the user can merge the task hierarchies of different task recipients to construct an "optimal" best-practice for the delegated task. The finally selected or constructed best-practice can be set as a suggested task pattern for further reuse. Task patterns for unused recipients' tasks and the related associations in delegation entities can be removed. Finally, the requester task contains delegation entities, comprising only recipients' information for all recipients, and one suggested task pattern to be used by all recipients in future process executions.

5.2.2.3.2 Delegation of One Generic Task to Multiple Recipients

A delegation of one task to multiple recipients may further denote that different persons have been asked to do the same thing collaboratively. For example, a chief executive officer asks several department managers to coordinate and prepare a whitepaper for a new product line. Thereby, different departments may need to take care of different aspects of the document, i.e. one should prepare the graphical layout of the paper, another product features etc.

The correct way to handle such business case is to decompose the whitepaper task in advance into sub-tasks, which reflect the different facets of the global task. These sub-tasks would then be delegated to the respective departments' managers according to their areas of expertise. If however a single, generic task is delegated, the resulting lack of structure can be corrected during the task pattern extraction. Correction can be performed in that the person extracting the task pattern for whitepaper preparation (e.g. the chief executive officer) is able to trace how the generic delegated task has been processed by all different recipients.

Different decomposition of the accepted tasks is expected by the different department managers as the delegated tasks refer to different operations from business perspective. These differences can be reflected in the extracted task pattern, by adding department-specific sub-tasks in the generic whitepaper task, and transferring the delegation associations for the respective recipients (department managers) to these sub-tasks. Each of the sub-tasks eventually has one delegation entity, containing only recipient information, and one suggested task pattern pointing at the respective recipient's task of the associated department manager. Hence, enabling a single suggested task pattern for a given collaborative task requires refinement of a captured task pattern towards correct task assignment based on self-contained tasks.

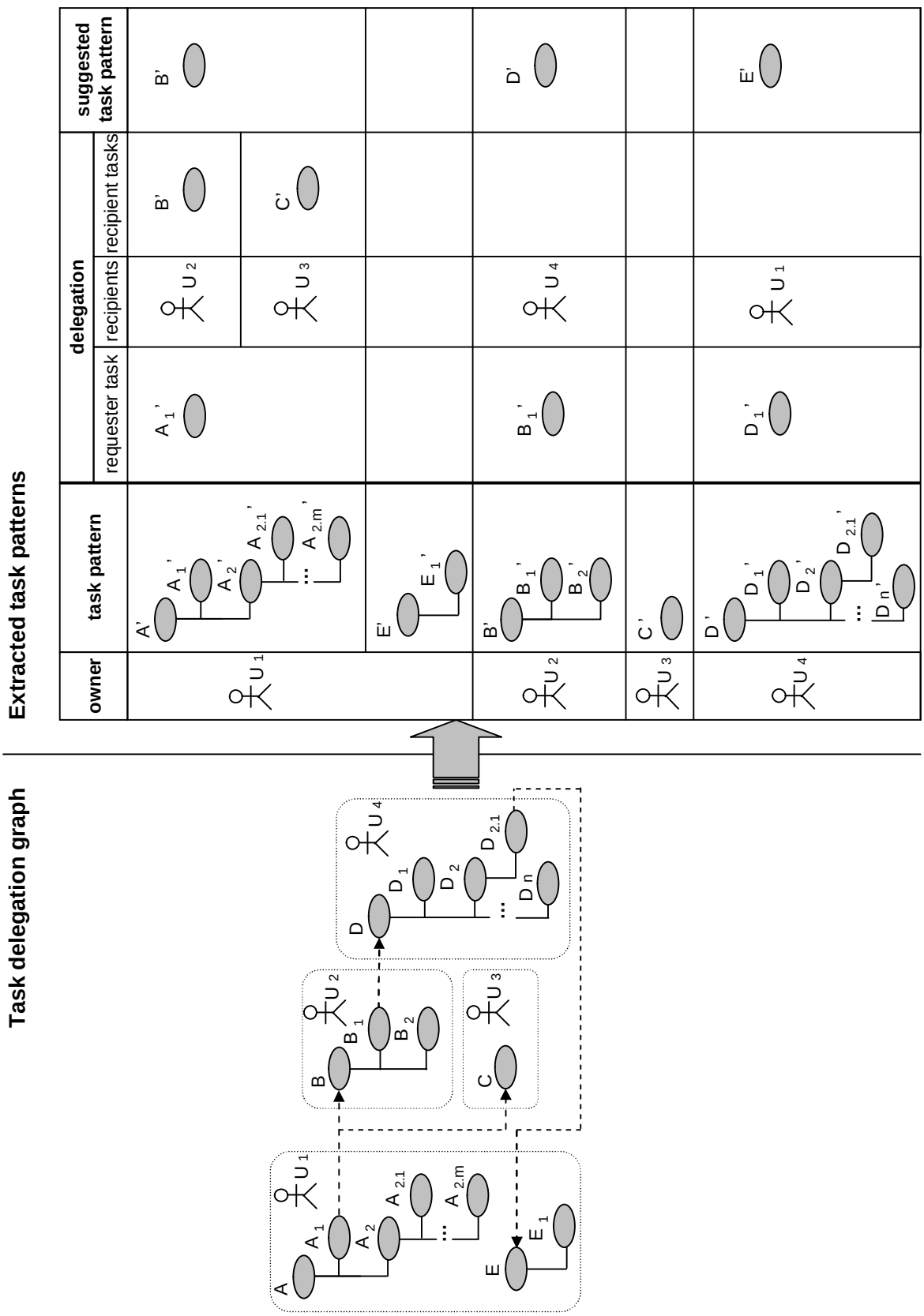


Figure 5.3: Decomposition of a task delegation graph into task patterns

5.2.2.3.3 Delegation to a Single Recipient

In case of task delegation to a single recipient, the requester task in a task pattern receives a single delegation entity, containing only recipient information, and a suggested task pattern. In Figure 5.3 task *B'* receives only user information for the recipient U_4 and a suggested task pattern reference to task *D'*. The important aspect here is that decoupling suggestion for further task handling from concrete recipient information enables changing the recipients and the suggestion independently from one another. This enables flexible adaptation of the declared best-practice.

5.2.3 Task Pattern Editing

The decomposition of task delegation graphs during task pattern extraction has leveraged the importance of providing flexibility and different interpretation of captured (tracked) process execution examples. Thus, enhanced task pattern editing is considered for adaptation and reconciliation of best-practices. An important concept for the adaptation of task patterns as captured process execution examples is to enable their editing through direct manipulation where *"the user is not required to interact in the interface domain of computational abstraction, but works directly with the data that interests him or her"* [Bla06]. Therefore, the thesis suggests that task pattern editing should be enabled through directly accessing human-readable task data, with which end users are familiar from their task management activities in the local workspace.

Task attributes such as name and description can be altered through common text editing.

Human actor associations can be edited based user information which is available in the selected integration environment. For example email clients such as Microsoft Outlook provide address book data which can facilitate the editing of user associations.

Artifacts adaptation for externally-managed artifacts is performed by explicitly searching, editing and attaching such artifacts according to the access policy of the respective artifact repository or document management system. If the users are not familiar with such systems, using externally-managed artifacts can be hindered due to the required effort for explicit document management. Externalized artifacts can be added in task patterns as common attachments in the same manner as in task instances. These artifacts are then implicitly replicated to the artifact repository. Thus the usage of externalized artifacts is rather unobtrusive. Locally-managed artifacts can be added only to local task patterns. If such artifacts need to be associated to a global task pattern, they need to be transformed to one of the other artifact types.

Suggested task pattern references to global task patterns can be set in both - local and global task patterns. Local task patterns on the other hand cannot be accessed from the global scope. Therefore suggested task pattern references to local task patterns are only possible from other local task patterns which have access to them in the local scope. Setting of suggested task patterns on user interface level can be supported through copy/paste or drag and drop functionalities.

5.2.4 Structural Adaptation of Task Patterns

Structural adaptations of task patterns are enabled to allow decomposition and reuse of task hierarchies at different level of details.

Adding and removal of tasks in task pattern hierarchies are considered as basic operations for structural editing. *Adding* of tasks extends existing task pattern structures. During *removal* all suggested task pattern references to the removed tasks in other task patterns are also removed. Existing references can be evaluated before the remove operation to allow the user to estimate whether the deletion will cause inconsistencies in other task patterns.

Moving and replication of task patterns have various implications related to the local and global task pattern scopes. *Move* can be realized through different operations such as cut/paste or drag and drop. Moving has implications in three major aspects: (i) moving a task pattern (root task) into another task as a sub-task; (ii) moving a sub-task on root level as a separate task

pattern; (iii) changing the scope of a task from local to global or vice versa. The latter aspect is relevant also for the replication of task patterns. *Replication* can be performed through copy/paste operations on task patterns. A replicate (copy) of a task inherits all context attributes, associations to artifacts, human actors, delegation entities and task structure through iteratively creating replicates of the sub-tasks in the hierarchy.

Moving of a task pattern as a sub-task for another task means that the task pattern is no more generic and is relevant only for the specific case of the target parent task. For example, Figure 5.4 (a) shows three task patterns on root level: "Organize workshop", "Organize Christmas party" and "Organize food", where the latter task pattern is referenced (dotted arrows) as a suggested task pattern in the respective "Organize food" tasks of the first two task patterns.

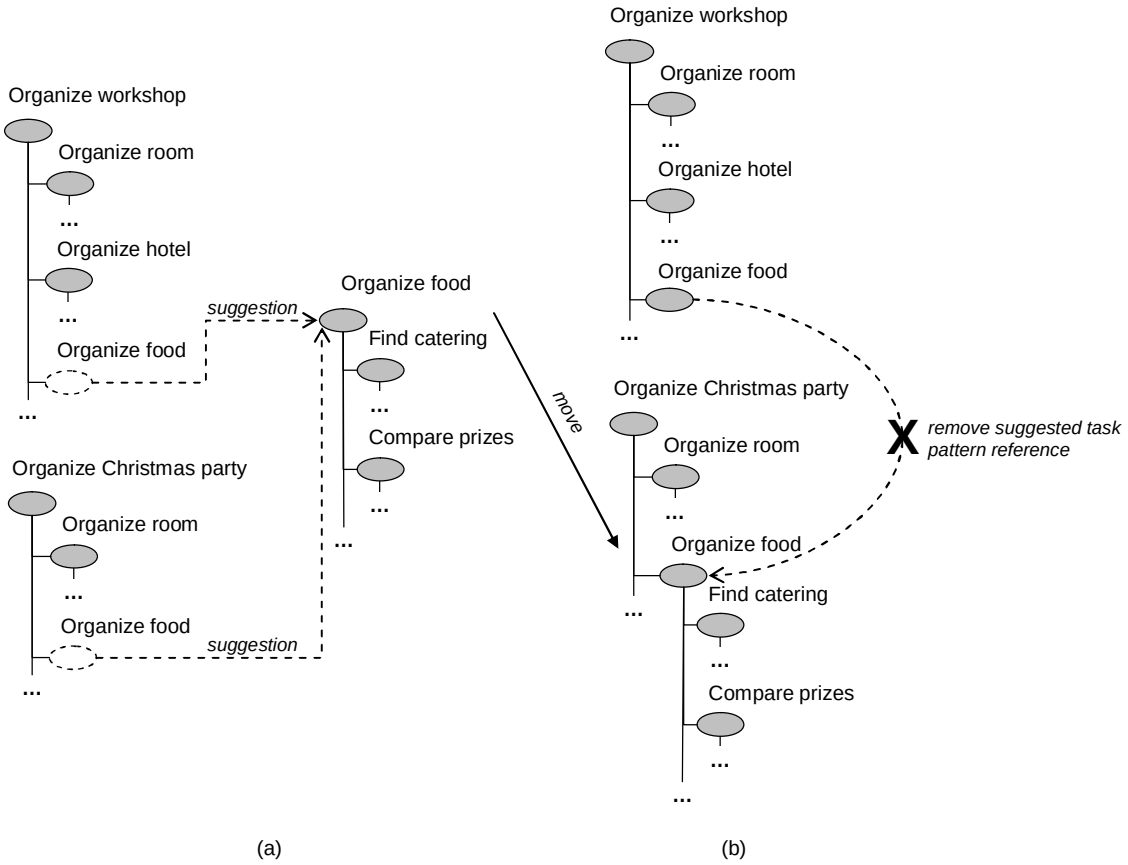


Figure 5.4: Moving task pattern as sub-task into another task pattern

An office manager assistant decides to move the "Organize food" task pattern as a sub-task into the "Organize Christmas party" pattern, by additionally removing the existing "Organize food" task in the latter pattern (Figure 5.4 (b)). This means, that the "Organize food" task pattern is now only applicable to the Christmas party task pattern, e.g. because changes in the internal regulations have occurred for ordering food for workshops with external partners on the one hand, and for internal events on the other. Suggested task pattern references to the moved task pattern are removed as this pattern is now made specific for a given business case. Before the move operation is performed, the user is prompted if they wish to remove all suggested task pattern references to the moved task ("Organize food") by viewing the referencing tasks to check if the moved task pattern is still applicable to some of them. If the moved task pattern is still applicable for multiple business cases, it can be preserved on root level to keep the suggested task pattern

references. Instead, the suggested task pattern references to this task pattern can be removed only from those tasks, for which the pattern is no longer relevant. For example, if the "Organize food" task pattern is referenced from a corresponding task also in a "Organize team building event" task pattern, the "Organize food" task pattern will be kept on root level to preserve the suggested task pattern references from the task patterns for organizing internal events (team building and Christmas party), whereas the suggested task pattern reference in the "Organize food" task of the "Organize workshop" task pattern will be removed. The "Organize food" task of the "Organize workshop" task pattern can be then refined with additional sub-tasks and other relevant information that relates to the specific business case.

Moving of sub-task (hierarchy) to root level as a separate task pattern on the other hand implies generalization and allows adding suggested task pattern references to the moved task. Following the above example, if the "Organize food" task is available as a sub-task in a task pattern for organizing a Christmas party, but the procedure for ordering food is applicable also for organizing a workshop, the "Organize food" task can be moved to root level as a separate task pattern and declared as suggested task pattern also in the task pattern for organizing a workshop.

Moving or replication of a task pattern from the local to the global scope requires adaptation of associated locally-managed artifacts and of local references to suggested task patterns. Locally-managed artifacts can be converted to one of the global artifact types or removed. Suggested task pattern references to local task patterns are handled as follows:

- iteratively move all suggested local task patterns to the global scope
- create copies for all suggested local task patterns in the global scope and switch the references to these copies
- remove the suggested task patter references to local task patterns

Suggested task pattern references to a moved task pattern in other task patterns from the original local scope are removed.

Moving or replication of a task pattern between different local scopes depends on the implementation of the local task pattern repositories. The thesis basically considers that different local task pattern repositories are not mutually accessible. In this case, if a local task pattern is moved/replicated between different local task pattern repositories, the references of this task pattern to local task patterns are handled in the same way as if the pattern is moved/replicated to the global scope, except that the suggested local task patterns are moved or replicated to the new local scope. Suggested task pattern references to a moved task pattern in other task patterns from the original local scope are removed.

Moving or replication of a task pattern from the global to the local scope preserves all artifact associations and suggested task pattern references of the moved/replicated task. When a move is performed, all global task patterns that have suggested task pattern references to a moved pattern lose these references because after the move the task pattern is not globally accessible. The task pattern editing environment can identify all such references and prompt the user to evaluate whether they really want to change the visibility scope of the task pattern, considering the implications from that for referencing patterns. The editing environment may not be able to evaluate suggested task pattern references to the moved global task pattern in all local task pattern repositories depending on the implementation of these repositories. Hence, such move operations can lead to inconsistencies in references from local task patterns to suggested global task patterns.

5.2.5 Task Pattern Exchange

Exchange of best-practices (R4) towards SER [FGY+04] of user-defined, weakly-structured process models is addressed in the thesis through different possibilities for exchange of task patterns – *implicit* and *explicit*.

Implicit exchange takes place through: (i) global availability of captured process execution examples (task delegation graphs) in a runtime, tracking repository; (ii) publishing of explicit

shared, global task patterns. In both cases search functionality is considered, allowing users to look for, inspect, extract and reuse previous experience, thus *pulling* task patterns from the available repositories. In the first case, the captured process knowledge is available implicitly in the tracking repository. The respective task structures can be retrieved in the form of task patterns. In the second case, task patterns are available as explicit best-practice definitions in global, shared manner. The patterns can be then reused directly.

Explicit exchange is performed by end users for through sending task patterns or links to global task patterns in email messages for explicit guidance. Such exchange is considered as important in the thesis because evidences show that many users approach their colleagues for help prior to looking for solution in the available software infrastructure [RJS02]. Explicit guidance is especially relevant for task requests, where task patterns can be sent as a proposition of how the requested task can be handled further. Thus, explicit exchange enables *push* behavior, where users are providing task patterns to each other rather than looking for these in the system.

5.2.6 Task Pattern Reuse

Reuse of best-practices (R4) towards SER [FGY+04] of user-defined, weakly-structured process models is enabled through application of task patterns on task instances in the user's to-do list in the local workspace. The application of a task pattern reactivates the captured or explicitly defined process example by generating the complete task hierarchy and filling all pre-modeled structure and content information in the to-do list.

5.2.6.1 Transfer of Task Context Information

Task attributes in textual form such as subject and description are transferred from task patterns to task instances in a straightforward manner. Runtime attributes such as priority, start date, due date, status and percent complete need to be set for the task instances explicitly, as these are not part of the task model that is provided by an applied task pattern.

Artifacts are loosely-associated to task instances and patterns and are managed in a similar manner in both task entities. Thus, artifacts can be transferred directly from task patterns to the respective task instances.

Human actor information is transferred only in terms of suggested delegation flow for collaborative tasks. The delegation entities of task patterns are interpreted during task pattern application and create *suggested recipient* associations for task instances (cf. Figure 4.2). If a user initiates a delegation for a task instance with suggested recipients, these are proposed automatically to facilitate reuse of the previous experience. The user can change the anticipated (example) delegation flow by entering different recipients. Thus, flexible execution of a captured process example is enabled based on recommendations rather than on strict prescriptions.

5.2.6.2 Transfer of Suggested Task Patterns

When a task pattern is reused, suggested task patterns are associated to the resulting task instances (cf. Figure 4.2). Through this association a suggestion for further task handling is provided which enables knowledge exchange in two major directions. On the one hand, the person that has applied the pattern can follow the suggestion in a task instance to decompose it further, even if the task has been delegated in the original ad-hoc process from which the pattern was extracted. On the other hand, when a delegation is issued, the suggested task pattern information is embedded in task requests (cf. Table C-1). Suggestion information is further embedded in the resulting accepted task instance at recipient site. The recipient is able to inspect and reuse the suggested task pattern to further decompose and delegate sub-tasks, thus unfolding the collaborative process according to the provided example flow.

When a local task pattern with suggested task pattern references to other local task patterns is applied, the user is enabled to publish the suggested local task patterns to the global scope. This

replication makes the suggested task patterns accessible to all users and allows unfolding the task delegation graph according to the global best-practice definitions.

5.2.7 Facilitating Task and Process Analysis in the Context of SER

For enabling tracing of relationships between running processes and best-practices (R5) and tracing of best-practice evolution (R6) the thesis suggests establishing of ancestor/descendant relationships. These relationships are established in three operations: (i) extraction of task patterns from task instances, (ii) application of task patterns on task instances, (iii) replication (copy) of task patterns. The ancestor/descendant relationships depend on the task pattern scope (local or global). The relationships for the different task pattern scopes are clarified in the following by referring to Figure 5.5. The figure shows the hierarchy of a task instance *A*, which has been developed in the local workspace of user U_1 and tracked to the central server instance. Task *A* is extracted to a task pattern *A'*, and reused by producing a task instance *A''*.

During extraction of global task patterns from a task delegation graph, ancestor/descendant relationships are established iteratively between each task in the global task pattern hierarchy and the respective task instance from the task delegation graph, i.e. *A* is set as ancestor for *A'* and *A'* is added as descendant of *A*, A_1 is set as ancestor of A_1' and A_1' is added as descendant of A_1 etc. The ancestor/descendant relationships for task *A* are shown in Figure 5.5, table (a). The provided ancestor/descendant relationships are task entity based. Thus, the evolution of each task is traceable independently of the current hierarchy that contains it. Even if structural changes occur during task pattern editing of *A'* or during structuring of the task instance *A* in the to-do list, the user is able to see from what previous (ancestor) hierarchies a task was part or in what subsequent (descendant) hierarchies it has been reused.

During application of a global task pattern, the resulting task instances receive iteratively ancestor associations to the respective tasks in the task pattern, and the latter receive corresponding descendant associations. This procedure is analogous to the ancestor/descendant association during global task pattern extraction but the associations are reversed, i.e. tasks from the global task pattern are set as ancestors and the produced task instances as descendants. The relationships for task *A* are shown in Figure 5.5, table (a) where a new task instance *A''* is derived from the task pattern *A'*.

During extraction of local task patterns from a task delegation graph each task in the local task pattern receives a local ancestor association to the respective originating task instance. The consideration thereby is that local task patterns may allow access to the tracking repository to inspect the originating ad-hoc processes. The ancestor association can be based on the identifier of the respective originating task instance. Local task patterns are inaccessible from the global scope. Therefore, during extraction of local task patterns, a task instance does not receive a descendant relationship to the corresponding task in the task pattern. The ancestor/descendant relationships for the extraction of task *A* to a local task pattern are shown in Figure 5.5, table (b).

During application of a local task pattern, the resulting task instances receive iteratively ancestor/descendant associations to the respective originating tasks from the tracking repository which are stored as ancestor associations in the local task pattern, i.e. task *A* is set as ancestor for *A''* and *A''* is added as descendant of *A* instead of *A'*, A_1 is set as ancestor of A_1'' and A_1'' is added as descendant of A_1 instead of A_1' etc. The ancestor/descendant relationships for task *A* are shown in Figure 5.5, table (b). The ancestor/descendant relationships during application of a local task pattern are established between task instances from different ad-hoc process instances, whereas the local task patterns play a mediating role by storing locally information about the original task instances. An important remark here is that ancestor/descendant associations to task instances from the tracking repository can be established, only if the executed, past process instances are still available in the repository. Further, no ancestor/descendant relationships are established during the application of a local task pattern if it has been created from scratch as explicit best-practice definition.

Figure 5.5: Ancestor/descendant relationships

Replication of a global task pattern to the global scope sets iteratively ancestor/descendant relationships between the originating and the resulting global tasks.

Replication of a global task pattern to the local scope sets iteratively ancestor relationships in the resulting local task pattern tasks, pointing at the originating global tasks. Descendant relationships are not set as local task patterns are disconnected from the global scope.

Replication of a local task pattern to the global scope produces ancestor/descendant associations to tasks from the tracking repository if the local task pattern has been extracted from a process instance. If on the other hand the local pattern has emerged through replication of global task pattern(s), ancestor/descendant relationships to the originating global task pattern(s) result.

Replication of a local task pattern to the local scope transfers to the created task pattern (copy) available ancestor associations to tasks from the tracking repository in case that the replicated task pattern has been extracted from a process instance. If the local pattern has emerged through replication of a global pattern, ancestor associations to the originating global task pattern are transferred to the created task pattern replicate.

5.2.8 Limitations of the SER Capabilities

Extraction, editing and publishing of global task patterns, as well as ancestor/descendant analysis can require certain authorization mechanisms and can depend on the access policy for contained tasks, artifacts and human actors' information. Such considerations are not made here, as the thesis focuses end-user driven business process composition in small, collaborative groups. Thus, possible implications from organizational constraints on the extraction, adaptation and reuse of best-practices have not been considered.

5.3 Scientific Achievements

This chapter has presented a method that defines the functional flow on the task management model entities and enables the composition, management, extraction, adaptation and reuse of weakly-structured process models by end users. The method defines the binding of ad-hoc task instances to overall task delegation graphs as well as the pre-processing and post-processing of email messages for exchange of tasks and deliverables. The exchange of email messages follows the language/action perspective [Win98] according to the introduced task management model.

Proactive user involvement in process composition based on collaborative events is enabled through embedding of task-specific meta-information in email messages. The embedding of meta-information in the common email infrastructure follows the concept of computational email [Bor92] and aims at enabling unobtrusive process support. Namely, the user is able to act from their email application and is not required to switch to proprietary environments in order to handle collaborative events on tasks. Unlike related work on email-based workflows [ADMG97] the thesis suggests embedding of task-specific meta-information rather than of active software objects in emails. On the one hand, this meta-information enables context-specific actions on task-related messages in the email application environment. Through this the user is enabled to act according to the provided collaborative event and to control the process emergence. User control is intrinsic for enabling *"informed participation"* [FGY+04] of end users in business process composition. On the other hand, embedded meta-information is interpreted by the process composition environment to update the emerging process model in a centralized manner. Centralized handling is intrinsic for estimating the impact of state and structural changes in task delegation graphs and for realizing adequate notifications.

Concepts for the runtime behavior of user-defined, ad-hoc task hierarchies and for handling state and structural changes in ad-hoc tasks are not provided in related literature on user-centric support for ad-hoc processes [Ber00, Jor04, HMBR05, HRD+06]. Related task

management models [GOR+07] focus on structural relationships and on entities for supporting ad-hoc tasks but do not discuss the effects of task state changes in interrelated ad-hoc tasks of different users. The method for composition of weakly-structured process models defined in the thesis provides mechanisms for handling state and structural changes that affect task hierarchies of a single user and of multiple users throughout a task delegation graph, i.e. changes respectively in the local and global scopes of a task delegation graph. Local and global scopes are considered important for group processes [ADML03], whereas an extensive discussion on how to deal with these scopes during dynamic changes in ad-hoc task hierarchies is not provided in related literature. Thus through the discussed mechanisms for tailoring operations in task delegation graphs, the provided method delivers novel concepts on ad-hoc process management based on user-defined task hierarchies.

Concepts for the transition between ad-hoc task instances and task patterns are not discussed in related work on task patterns for ad-hoc process support [RRMvdA05, GOR+07]. Task patterns are seen as a promising alternative for supporting ad-hoc processes as opposed to structured workflows [RRMvdA05]. Furthermore, task patterns are a concept that pertains to user-centric process support through capturing, exchange and reuse of process knowledge. Thus this concept addresses the *"seeding, evolutionary growth, and reseeding"* (SER) [FGY+04] process model for user-tailorable systems from task management and business process management perspectives. Related task management models [GOR+07] leverage task patterns as a concept for user-centric process support. The method discussed in this chapter provides extension to the task pattern concept through concepts for the transition from runtime ad-hoc task instances to task patterns upon task pattern extraction, and from task patterns to task instances upon task pattern reuse. The method defines the decomposition of a task delegation graph into task patterns and the assembling of task patterns upon reuse towards unfolding of complete task delegation graphs. Through this also the concept of programming by example [Cyp93, Lie01] of weakly-structured process models through capturing and repeated execution of ad-hoc business processes is refined.

Local and global scopes for task pattern definition, exchange and reuse are further defined through the presented method. The defined scopes result in constraints for explicit tailoring operations on task patterns. The discussed task pattern scopes further affect the ancestor/ descendant relationships, which are proposed in the method for enhanced case analysis in the context of SER. The provided method defines how the evolutionary ancestor/descendant relationships are established between task patterns and instances in the different scopes. Related literature on task patterns for ad-hoc process support [RRMvdA05, GOR+07] does not consider local and global task pattern scopes and their effect on explicit tailoring and exchange of task patterns, or on evolutionary ancestor/descendant relationships. Thus the provided method extends the overall task pattern concept by defining the relationships between personal and group best-practice definitions and the transitions between both types of best-practice definitions.

5.4 Summary

This chapter has presented a method for composition of weakly-structured process models based on collaborative task management. The method defies the functional flow for the entities from the introduced task management model and how end-to-end processes are aggregated from the underlying user applications for task management (to-do lists) and collaboration (email).

The creation and replication of task instances has been discussed for constructing personal task hierarchies. These hierarchies are bound on a central server instance through tracking the email exchange for task delegation. Support for collaborative task handling and assembling of task delegation graphs is provided through embedding task-related information in conventional

email messages. The emails for exchange of tasks and deliverables are based on generic speech acts. Basic notifications for changes in a task delegation graph have been further discussed.

The extraction, adaptation and reuse of user-defined, weakly-structured process models through task patterns have been presented. Extraction and reuse are supported through transitions between task instances and task patterns. Decomposition of task delegation graphs into task patterns and reconciliation of task delegation graphs from task patterns based on recommendations have been discussed. Editing of task patterns has been discussed with respect to local and global visibility scopes. Evolutionary ancestor/descendant relationships upon reuse of task hierarchies have been presented, which enable enhanced case analysis in the context of SER.

CHAPTER 6: A Method for Composition of Structured Process Models

The method discussed in this chapter extends the conceptual framework of the thesis with capabilities to derive structured workflow models from weakly-structured task delegation graphs. Such derivation enables a seamless transition from user-defined to formal process models towards automation of rigidly recurring processes (R7). The introduced method further enables extension of structured workflow models based on deviations in workflow instances through ad-hoc tasks. The method founds on the task management model introduced in Chapter 4.

6.1 From Email and To-Do to Formal Process Models

The previous chapters have described a task management model and a method for the composition of weakly-structured process models based on personal task management. Weakly-structured process models are composed dynamically in the form of task delegation graphs through definition and hierarchical task decomposition of explicit task representations in light-weight to-do lists, and through task delegation over email. Task patterns can be extracted from task delegation graphs and reused as process examples that abstract from a specific process instance to reconstruct ad-hoc business processes for recurring cases. Through this capturing and repeated execution of ad-hoc processes, programming by example [Cyp93, Lie01] of weakly-structured process models is enabled on enterprise level. The composition of weakly-structured process models requires manual activities for managing ad-hoc task instances and task patterns and does not support process automation (R7).

Process automation can be supported through conventional workflow management systems based on structured process models [vdAvH02, vdAHW03]. Additionally, a need has been discussed (cf. Table 3.1, Section 3.1.1.5) to integrate the business and technology perspectives on business processes by enabling process tailoring as collaboration between end users, process designers and developers [MM00]. Task delegation graphs are based on hierarchical task decomposition and delegations (cf. also Definition 4.1). Additionally, a task instance change history is maintained for each ad-hoc task instance as discussed in the task management model (cf. Section 4.4.2). The change history of an ad-hoc task instance captures various stages in the elaboration of that task instance and allows evaluation of temporal relationships between ad-hoc task instances in the course of an ad-hoc process instance. Thus, the structural relationships between tasks in a task delegation graph and the task instance change history can be used to derive control flow for ad-hoc task instances. The discussed task management model further provides artifact (cf. Section 4.6) and human actor (cf. Section 4.7) associations to ad-hoc task instances. Hence, process automation can be enabled through transformation of a task delegation graph to a structured workflow model by using user-defined information on task flow, document flow, and involved human actors in terms of task assignments from a real-life process execution in a task management system. Such transformation can additionally deliver a shared context for process model interpretation and tailoring between end users, process designers and developers.

Hierarchical task decomposition is proposed by a large body of research for supporting ad-hoc processes [ADMG97, Ber00, Sch03, HRD+06, RRMvdA05, GOR+07, Sch09]. However, methods for transformation of user-defined, ad-hoc task hierarchies to structured workflow models are not discussed in related literature. [Ber00] discusses integrated ad-hoc and procedural process support that is based on light-weight task hierarchies. For process automation [Ber00] suggests using imperative workflow scripts that direct the execution of tasks whereas the overall process remains based on task hierarchies. [ADMG97] presents a system, where automation is

supported through explicit process modeling in a simplified visual notation without considering transformation of hierarchical task structures from a provided to-do list to a workflow model.

On the other hand, workflow management and business process management studies focus on embedding flexibility in structured workflows [vdABV+99, Jor04, vdAWG05, Ber05]. The latter studies consider having an initial, preliminary workflow model and do not discuss derivation of formal models from ad-hoc task hierarchies. The lack of conceptual work on the transformation of ad-hoc processes or process fragments in the form of task hierarchies to structured workflow models motivates the introduced method. The method discusses the transformation of task delegation graphs to structured workflow models by focusing on three major aspects: control flow, document flow, and human actors' information in terms of task assignments.

6.2 Control Flow Transformation

The major difference between task delegation graphs and workflow models is that the first provide structural decomposition and delegation flow of tasks whereas the second specify the control flow in processes, i.e. the sequence in which tasks are executed. To clarify the transformation of task structure to task sequence flow, the thesis first provides a brief introduction to the relevant terminology for workflow graphs, and introduces some basic terms related to the transformation of task delegation graphs.

6.2.1 Terminology

Workflow graphs are based on two-terminal graphs [VVK08]. A two-terminal graph is a directed graph G [Die00] such that there is a unique source node s and a unique sink node $t \neq s$ and each node v is on a directed path from s to t. The thesis considers a workflow graph as a series-parallel graph with distinguishable node and edge types, where branching is modeled through gateways in a block-oriented fashion (cf. also [RD98, RRD03]). The thesis further considers only entities and relationships that can be derived from a task delegation graph and defines a derived workflow model as follows.

Definition 6.1: (Derived Workflow Model) A derived workflow model is a tuple $S = (V, D, VT, CtrlE, DataE)$ where:

- V is a set of tasks and D a set of process data elements
- $VT: V \mapsto \{StartFlow, EndFlow, Task, AndSplit, AndJoin, XOrSplit, XOrJoin\}$
- $CtrlE \subset V \times V$ is a precedence relation
- $DataE \subseteq V \times D$ is a set of data links between tasks and data elements

In the above definition the term "task" is used as a synonym for "workflow task model", i.e. a workflow model is a model of an operational business process, which is composed of workflow task models as fine-granular building blocks (cf. Definitions 1.4 and 1.5). Workflow tasks models in a workflow model result from task instances in a task delegation graph, and process data elements from artifacts. The definition does not include entities that cannot be derived directly from a task delegation graph such as looping. After the transformation of a task delegation graph to a workflow model, the process modeler is able to extend the derived workflow model with additional elements according to the concrete formal modeling notation. A *workflow graph* is the graph representation of a derived workflow model. A workflow graph is correct if (cf. [RRD03]):

- for each split node (fork) there is a unique join node
- S is structured following a block concept, i.e. control blocks (sequences, forking) can be nested but must not overlap

The **block** concept plays an intrinsic role in the transformation of task delegation graphs to structured workflows. A block in a workflow graph is a hierarchy of sub-workflows that have a single entry and a single exit of control [VVK08]. The notion of block introduced by the latter study is adopted in the thesis, where a block is considered as a connected sub-graph with unique entry and exit *nodes*. For the transformation of task delegation graphs to workflow graphs, a block is considered as a hierarchy of sub-workflows that is derived from a set of ad-hoc task instances and represents a part of the structured workflow model.

An **evaluation set**, contains all task instances produced through ad-hoc task management, which are evaluated to form a *block* in the workflow model. The transformation of a task delegation graph to a workflow model is performed through assembling evaluation sets and producing respective task sequence blocks - first for the root task, and then iteratively for all tasks with sub-tasks throughout a task delegation graph. Depending on the interpretation of task decomposition and delegation flow, evaluation sets can comprise tasks from different parent tasks, accepted tasks or ad-hoc processes as discussed throughout this chapter.

An **associated evaluation set** for a given task from the task delegation graph, is an evaluation set that is used to generate the workflow block containing the corresponding workflow task node for this task. Recall that according to the runtime task management model (cf. Figure 4.2) a task instance can have an associated workflow task model. In the following, this workflow task model is the task node that is produced from the task instance in the generated workflow graph.

A **task delegation graph** in the following refers to the ad-hoc process fragment that has been selected for transformation. The thesis suggests that a user should be able to initiate workflow transformation from an arbitrary task in a task delegation graph. This enables formalization only of those facets of an ad-hoc process, which a user finds appropriate for automation.

A **root task** in the following refers to the root of the task delegation (sub-)graph that is being transformed and does not necessarily coincide with the root task for an overall ad-hoc process.

An **atomic task** is a task from a task delegation graph, which has no sub-tasks. A task from a task delegation graph that has sub-tasks is referred to as *non-atomic* task or as *parent* task.

An **initial task** is a task from a task delegation graph which can be interpreted differently during process model transformation. Tasks that contain sub-tasks are marked as initial tasks when their parent task is processed as discussed later in this chapter. The final type of an initial task in the workflow model is determined in a subsequent transformation step.

An **initial node** is a node in the workflow graph, which represents an initial task from the task delegation graph. The final type of an initial node is determined in a subsequent processing iteration, i.e. when the corresponding initial task is transformed.

A **target node** during process model transformation is a node in a workflow graph where the workflow block from the next transformation step should be inserted.

A **target graph** is a workflow graph in which the generated workflow block from a transformation step should be inserted. The transformation of a single task delegation graph can result in multiple workflow graphs, representing a main process and multiple sub-processes as discussed further in the transformation method.

A **strict delegation sub-graph** for a given delegated task in a task delegation graph is the graph which encompasses the task, and iteratively all its recipients' tasks and their recipients' tasks excluding any hierarchical decomposition. A strict delegation sub-graph is shown in Figure 6.1 and defined formally as follows. Let $G\ (V, E)$ be a task delegation graph, where V is the set of all user-defined task instances, and $E = E_d \cup E_h$ is the set of all edges, with E_d being the set of edges representing delegations, i.e. connecting requester tasks with recipient tasks, and E_h being the set of edges representing hierarchical decomposition, i.e. connecting parent tasks with their respective sub-tasks. A sub-tree $G_A\ (V_A, E_A)$ with root node $A \in V$ is called a *strict delegation sub-graph* for A, if $E_A \subseteq E_d$ and $E_A \cap E_h = \phi$.

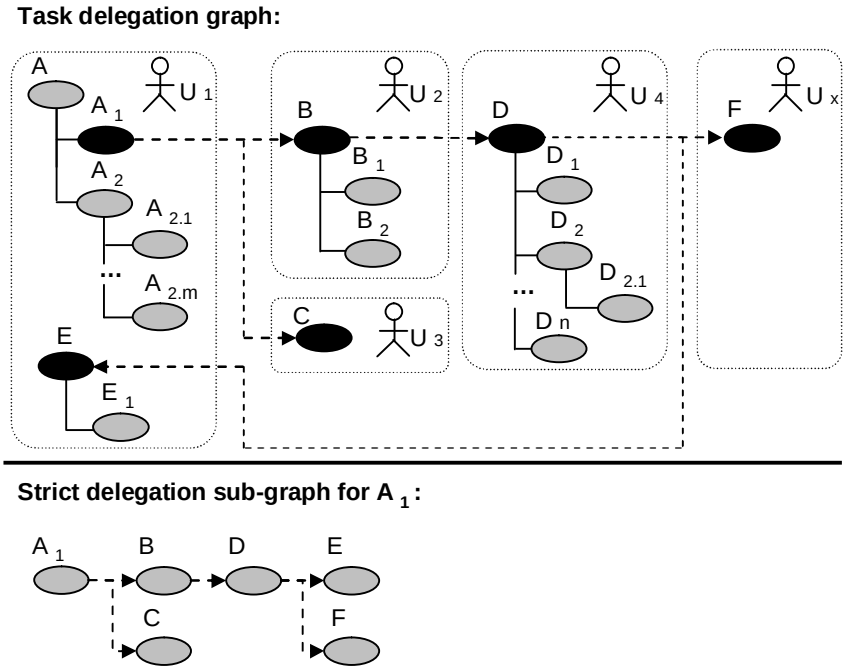


Figure 6.1: Strict delegation sub-graph for a task in a task delegation graph

6.2.2 Traversing a Task Delegation Graph

The thesis suggests enabling a step-wise, iterative transformation of a task delegation graph to a workflow model. Step-wise transformation enables users to better evaluate the reflection of user-defined process models into formal workflow graphs. This approach further allows derivation of workflow models with different granularity, i.e. the user can interrupt the transformation after only a set of high-level tasks in the hierarchy have been transformed. The generic algorithm for traversing a task delegation graph is provided in the following.

Algorithm 6.1: Traversing a task delegation graph for workflow graph generation.

Require: *G* is a task delegation graph.

1) Create a new empty workflow graph and set is as *target graph* for the transformation
2) Start at *root task* of *G*
3) Transform *task* from *G* to assemble *evaluation set* from sub-tasks and delegated tasks
4) Mark all *atomic tasks* detected during the assembling of the evaluation set as *processed* and all *non-atomic* tasks as *initial tasks*
5) Generate *block* from the *evaluation set* by: (i) creating *initial workflow nodes* for *initial tasks*; (ii) associating *task instances* to the corresponding derived *workflow task nodes*; (iii) associating *workflow task nodes* to the containing *workflow graphs*
6) Insert generated workflow *block* in *target graph* according to specified *target node*
7) Find next *initial task* in the *task delegation graph* in a breadth-first traversal

   If next *initial task* is found **then**

   Set corresponding *initial workflow node* as *target node*

   Set the graph containing the *target node* as *target graph*

   Repeat the algorithm starting at 3) with the found *initial task*

   Else

   The transformation of *G* is complete

The introduced generic algorithm is detailed in the remained of this chapter where the transformation of root tasks, initial (parent) tasks and the generation of workflow blocks are discussed. An example traversal of a task delegation graph is shown in Figure 6.2, i.e. according to the graph fragments that are transformed in the different traversal steps. The processing starts from the root task *A*, and continues with parent tasks in a breadth-first traversal. Each processing step assembles an *evaluation set*, which produces a workflow block. Different interpretations of hierarchical task decomposition and delegations result in different evaluation sets. These interpretations and the assembling of evaluation sets are discussed in the next sections.

A central point in the traversal procedure is that all tasks in a *strict delegation sub-graph* of a given requester task are considered as residing on the same level in the task hierarchy as the requester task, i.e. as being virtually children of the parent task of this requester task. Thus, tasks from the strict delegation sub-graph of a requester task are processed together with the requester task itself and reside in the same *evaluation set*. For example when *A* in Figure 6.2 is transformed, its sub-tasks A_1 and A_2 are processed together with the recipients' tasks of A_1 (tasks *B* and *C*).

During each transformation step, non-atomic tasks are marked as *initial tasks*. For example, when task *A* is transformed, tasks *B* and A_2 are marked as initial tasks. Initial tasks determine the target for the next transformation (traversal) step.

In case of delegations, requester and recipients tasks can be merged by selecting one of them as the preferred final task for the workflow model. In the latter case the sub-tasks of requester and recipients tasks are handled as children of the same parent, i.e. the selected preferred task. Thus, if any of the merged tasks has sub-tasks, the selected preferred task is marked as initial task even if it does not have sub-tasks in the task delegation graph. For example, when task *A* is transformed, A_1, *B* and *C* can be merged by selecting one of them as a preferred merge task *M*. If A_1 or *C* is selected as a merge task for *B*, the selected merge task is considered as initial task because *B* has sub-tasks. Merge is discussed in details in Section 6.2.4.3.

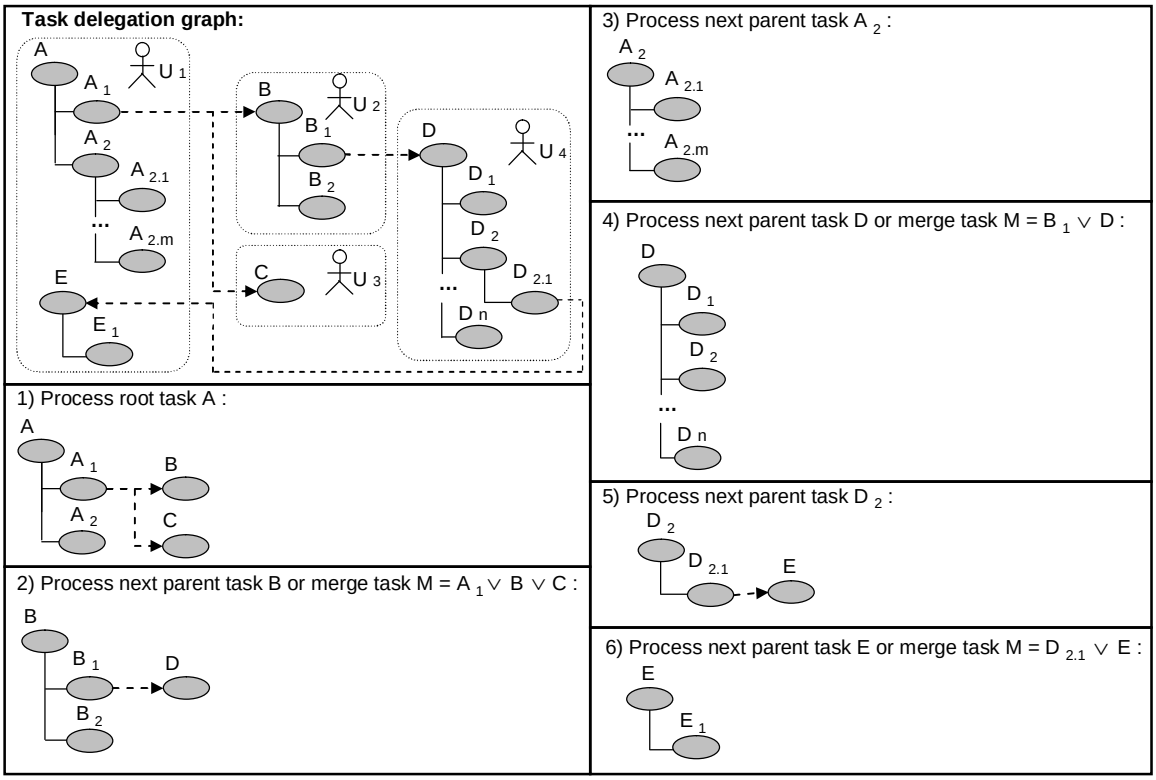


Figure 6.2: Example traversal of a task delegation graph

6.2.3 Interpretation of Hierarchical Task Decomposition

Hierarchical decomposition of tasks into sub-tasks enables end users to refine the working tasks and to specify them to a lower level of details. Decomposition of ad-hoc tasks in hierarchical to-do lists is largely considered in related literature on task management and user-centric process support [Ber00, HRD+06]. However, the transformation of hierarchical decomposition to workflow graphs is not discussed in the literature. The thesis proposes different interpretations of such decomposition for the transformation of task delegation graphs to structured workflows.

6.2.3.1 Parent Task to Sub-Process

Formal modeling notations enable modularization of process models through non-atomic sub-processes [OMG06]. On the one hand, this modularization provides simplification of the visual process models. On the other hand, modularization fosters reuse as sub-processes receive a single input and deliver a uniform output by performing a self-contained functionality [RM08]. Thus sub-processes can be reused as modules in different larger process models.

The thesis suggests enabling transformation of non-atomic tasks from a task delegation graph, to non-atomic sub-processes in a derived workflow model. This transformation is especially relevant if sub-tasks in a task delegation graph inherit some of the context attributes or artifacts of their parent task. Such inheritance would imply that the input of the parent task is distributed to the sub-tasks, similarly to the interpretation of sub-processes in formal process modeling.

6.2.3.2 Parent Task to Atomic Task

A parent task in a task delegation graph may have context attributes and artifacts which are not transferred to any of its sub-tasks. This can imply that the task decomposition in the task delegation graph is not correct in the sense that the sub-tasks do not represent single steps towards reaching the goal of the parent task. For handling such cases, the thesis suggests transformation of a parent task to an atomic workflow task, preceding the sequence of the sub-tasks.

Precedence is explicitly considered as the parent task cannot be completed before the sub-tasks during ad-hoc task management, i.e. completing the parent task completes also all sub-tasks (cf. Section 5.1.5.2). Thus no evaluation of the execution sequence of the parent task and the sub-tasks based on the tasks' change history can be performed as discussed further in this chapter.

6.2.3.3 Parent Task to Logical Group Association

Formal process modeling languages [OMG06] consider logical association of tasks for documentation and analysis of process diagrams. The provided group association elements [OMG06] do not affect the task flow. The thesis considers transformation of a parent task from a task delegation graph to a logical group association for the case that a parent task does not contain any task details in terms of textual description or artifacts. In this case, a parent task may represent a logical unit which roughly identifies the overall goal of a task and serves mainly for grouping the concrete sub-tasks.

The proposed transformation of a parent task to a logical group association considers two basic options: (i) *transformation with* and (ii) *transformation without merge of grouped sub-tasks at the hierarchical level of the initial parent task*. The transformation options are exemplified respectively in Figure 6.3 (a) and Figure 6.3 (b) and discussed in the following.

During parent task transformation to logical group association with merge of the grouped sub-tasks at the hierarchical level of the initial parent task (Figure 6.3 (a)) the grouped sub-task nodes $a_{2.1}$ to $a_{2.m}$ resulting from $A_{2.1}$ to $A_{2.m}$ are merged with the sub-task nodes resulting from the transformed parent task A. The only sub-task of A that is considered in Figure 6.3 is A_1 as A_2 itself is transformed to a logical group association L_{A2}. Transformation with merge allows evaluation of task sequence based on a common evaluation set comprising the sub-tasks of A and A_2 (see T in Figure 6.3 (a)). Thus, this transformation "flattens" the task hierarchy by considering the parent

task A_2 and its sub-tasks as residing on the same level. In this case the generated group element L_{A2} may encompass also workflow task nodes for sub-tasks of the previously processed parent task A such as a_1 depending on the detected task sequence. Sequence generation from an evaluation set is discussed further in this chapter.

During parent task transformation to logical group association without merge of the grouped sub-tasks at the hierarchical level of the initial parent task (Figure 6.3 (b)) the sub-tasks $A_{2.1}$ to $A_{2.m}$ of the processed initial parent task A_2 are comprised in a single evaluation set. The sequence of the processed sub-tasks ($A_{2.1}$ to $A_{2.m}$) is generated independently from the sequence of the sub-tasks (A_1 and A_2) of the previously processed parent task A. Thus the generated group element groups only workflow nodes ($a_{2.1}$ to $a_{2.m}$) that are generated from the sub-tasks of the transformed initial parent task A_2.

Independently of the transformation type a *group element association* is established on system level between each grouped workflow node and the corresponding logical group element, i.e. $a_{2.1}$ to $a_{2.m}$ receive a logical group association to L_{A2}. This association on system level is required to recover group elements when redefining generated workflow blocks. For example, let A_3 be a further non-atomic sub-task of A. Let A_3 be transformed to a logical group association through merge by producing a group element L_{A3}. After A_3 is transformed, L_{A2} has to be recovered so that no group associations from previous transformations are lost. Thereby L_{A3} and L_{A2} may (visually) overlap depending on the detected sequence of the tasks in the evaluation set.

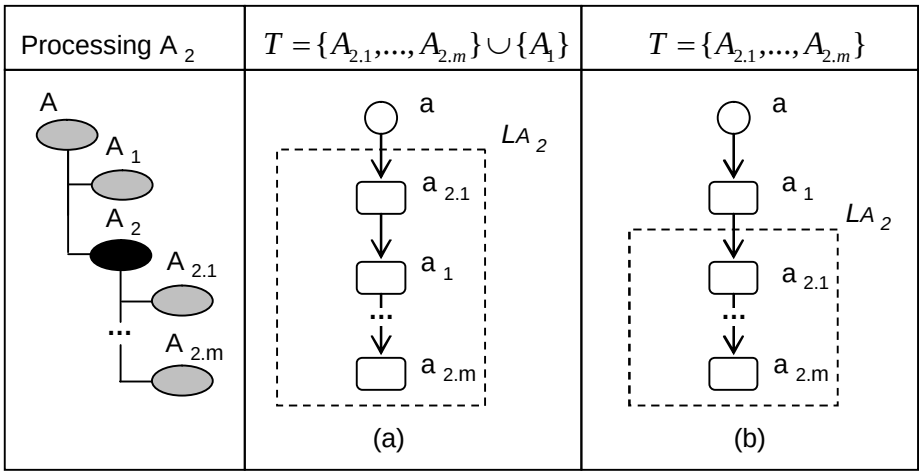


Figure 6.3: Transformation to logical group association - the transformed task is A_2, the assembled evaluation set is T

6.2.3.4 Omitting Parent Tasks

Omission of parent tasks during transformation of a task delegation graph is considered for simplification of the derived structured process model. Omission is relevant if a transformed parent task does not have context information and thus merely groups its sub-tasks, and if the user intends to explicitly declare logical group association other than the existing parent task, not to declare any group association at all, or if the workflow modeling notation selected for export does not allow grouping. The transformation of a parent task through omission is performed in the same way as export to a logical group association and considers the same two options (with and without merge at hierarchical level of parent task) except that no logical group element is created from the transformed parent task.

6.2.4 Interpretation of Delegations

Task delegation graphs represent execution examples for ad-hoc processes, which emerge in an ad-hoc, underspecified manner. The unplanned behavior may produce collaborative tasks which have different meaning from business perspective and which require different interpretation during process model transformation. The thesis proposes transformation of collaborative tasks through three different interpretations: (i) *omission*, (ii) *preserving*, and (iii) *merge*. These are discussed in the following sections. An important notice for the following discussion is that delegations can be performed iteratively and a requester task may have an involved strict delegation sub-graph (cf. Figure 6.1). The thesis suggests that during the process model transformation the user should be able to inspect all collaborative tasks in the strict delegation sub-graph of a delegated task and to estimate which tasks to omit, preserve, and merge.

6.2.4.1 Omitting Collaborative Tasks

Omitted tasks are excluded from the transformation procedure and do not produce workflow task models. Omission aims at simplification of the derived workflow model.

Omission of requester tasks is considered for the case that a delegated task in a task delegation graph is fully processed by the recipient(s). For example, a managing director has created a task for organizing a steering committee meeting which they have delegated to their assistance. The director is not involved in the task but is able to refer to the local task representation in their to-do list, and to switch to the global process overview to inspect the further processing of the task. In this case the requester task does not incorporate information about the actual processing of the task.

Omission of recipient tasks is considered if a recipient has accepted the task, but was unable to process the task for some reason and the task has been processed by the requester or by another recipient(s). Following the above example, the managing director has delegated the task for organizing a steering committee meeting to one of their assistants, who has accepted the task but was unable to process it on time, e.g. because of illness leave or other unexpected circumstances. Then the managing director has delegated the task to another assistant or processed it themselves.

A further use case for omission of recipient tasks may arise if the task assignment and distribution in ad-hoc processes differs from those in structured workflows. On the one hand, multiple recipient tasks may exist for a delegated task in a task delegation graph. On the other hand, in a workflow model a single task may be distributed to different stakeholders during workflow execution based on multiple task assignments. Thus, during process model transformation one of the tasks in the strict delegation sub-graph can be *preserved* for generating a single workflow task in the derived workflow and the other recipient tasks can be *omitted*. Such omission prevents from exporting multiple, redundant tasks to the structured workflow model but requires that after the transformation the user (manually) assigns the derived workflow task to the owners of omitted recipients' tasks. Omission further raises issues if some of the collaborative tasks have sub-tasks denoting low-level activities that need to be performed by the different stakeholders. These issues are addressed through the merge option as discussed later on.

6.2.4.2 Preserving Collaborative Tasks

All requester and recipients tasks that incorporate information about the actual task processing can be preserved during process model transformation. Requester tasks can be preserved if the requester task contains information about the task processing that is not contained in the recipient task(s) and the requester needs to perform on that task. An indication for preserving a requester task may be for example the availability of sub-tasks in the requester task. This case can arise if a user has delegated a task to other persons but later on noticed that they need to perform some activities on that task themselves and created sub-tasks for the already delegated task. Preserved tasks produce corresponding workflow task models. Preserving multiple tasks in a strict

delegation sub-graph may be unacceptable if the workflow model requires a single task node with multiple assignments, which is then distributed to multiple process participants during workflow execution. In this case the merge option can be applied.

6.2.4.3 Merging Collaborative Tasks

Merging can be applied to two or more preserved collaborative tasks from a strict delegation sub-graph. Preserved collaborative tasks are merged by selecting one of them as a *merge task*. More than one merge tasks can be selected if multiple collaborative tasks are preserved. A task from the strict delegation sub-graph is allowed to have only one merge task. The merge task cannot be a merge task to itself.

A task that has an associated merge task is referred to as *merged task*. A merged task cannot be selected as a merge task itself. Instead the associated merge task of that merged task can be used for merging. The latter rule applies as merge aims at consolidation of collaborative tasks during process model transformation and all merged tasks are considered as describing a common logical unit of work.

A merge task produces a single workflow task model for all merged tasks. Context information and assignments of multiple merged tasks can be transferred to the workflow task model that is derived from the associated merge task. The sub-tasks of the merged tasks are handled as sub-tasks of the merge task and comprised in a single evaluation set. This allows grouping collaborative tasks of different users in the same logical group association or sub-process. The sub-tasks of the merged ad-hoc tasks are preserved and transformed to workflow tasks, which are assigned to the corresponding various owners of the original merged tasks.

6.2.5 Task Transformation

This section provides the algorithms for transformation of a task delegation graph. Following the generic procedure for task delegation graph traversal, the discussion starts with root task transformation and then continues with initial (parent) task transformation. A root task is handled as a special case insofar it is the first transformed task in a task delegation graph.

The provided algorithms consider the discussed interpretation of hierarchical decomposition and delegations and clarify the assembled *evaluation sets*. An assembled *evaluation set* is the *associated evaluation set* for all tasks in it. *Merged tasks* receive the *evaluation set* of the associated *merge task* as an *associated evaluation set*. If *cancelled* ad-hoc tasks are encountered, the user is allowed to specify whether to include these in the evaluation set, or not.

Algorithm 6.2: Assembling of an evaluation set for a delegated task.

Require: A is a delegated task from a task delegation graph.

// Step 1. Select tasks for processing from the strict delegation sub-graph.

User selects which tasks from the *strict delegation sub-graph* G_A (V_A, E_A) of A to *omit* or *preserve*

Let $P_A \subseteq V_A$ be the set of all *preserved tasks*.

// Step 2. Merge tasks (optional).

If $|P_A| > 1$ **then**

User may perform merge by selecting a *merge task* for one or more *merged tasks* in P_A. User may further select which context information and assignments of *merged tasks* should be transferred to the resulting workflow node of the associated *merge task*.

Let $M \subseteq P_A$ be the set of all *merged tasks* ($M = \phi$ if no merge is performed).

// Step 3. Mark tasks as processed or initial to enable further task delegation graph traversal.

Mark all *merge tasks* that have an associated *non-atomic merged task* or are themselves *non-atomic* as *initial tasks*.

Mark all *atomic merge tasks* that have only associated *atomic merged tasks* as *processed*.
Mark all *atomic merged tasks* as *processed* and all *non-atomic merged tasks* as *initial tasks*.
Mark all *atomic tasks* $A_j \in P_A \wedge A_j \notin M$ that are not *merge tasks* as *processed* and all *non-atomic* tasks $A_k \in P_A \wedge A_k \notin M$ that are not *merge tasks* as *initial tasks*
Return $P_A \setminus M$ (i.e. the returned set encompasses all preserved tasks from the strict delegation sub-graph that do not have an associated *merge task* including *merge tasks* themselves)

Algorithm 6.3: Root task transformation.

Require: A is a root task from a task delegation graph.
// Step 1. Create workflow graph.
Create a new workflow graph G_w
Set *target graph* = G_w
// Step 2. Assemble evaluation set, mark processed and initial tasks and generate block.
If A is delegated **then**
Assemble *evaluation set* T from delegations by passing A to Algorithm 6.2
Generate workflow *block* from T
If workflow block starts with gateway **then**
Create a *dummy start node* in G_w and set it as *target node*
Insert the generated *block* after the *start node* in G_w
Else
Transform the first node from the generated *block* to *start node*
Insert the generated *block* in G_w
Else if A has a set of sub-tasks S **then**
Create *start node* in G_w from A and set it as *target node*
Initialize *evaluation set* T (to be filled in the following)
For each task A_s in S **do**
If A_s is delegated **then**
Assemble *evaluation set* T_d from delegations by passing A_s to Algorithm 6.2
$T = T \cup T_d$
Else
$T = T \cup \{ A_s \}$
If A_s is atomic **then**
Mark A_s as *processed*
Else
Mark A_s as *initial task*
Mark task A as *processed*
Generate workflow *block* from T
Insert the generated *block* after the *start node* in G_w
// Step 3. Create an end node to complete the workflow graph.
Create an *end node* and append it to the generated *block* in G_w

Algorithm 6.4: Initial (parent) task transformation.

Require: A is an initial task in a task delegation graph.
// Step 1. Determine the actual task for transformation and assemble evaluation set. Note that
// if A is an initial task it either has sub-tasks or is merged with other tasks that have sub-tasks.
Initialize $A_t = A$ as the actual task for transformation (to be determined in the following)
Initialize S_t as the set of sub-tasks for transformation (to be determined in the following)
If A has an associated *merge task* A_m **then**
Let M be the associated set of *merged tasks* for A_m ($A \in M$)

$S_t = S_m \cup S_1 \cup S_2 \ldots \cup S_n$ where S_m is the set of sub-tasks of A_m, and S_j ($1 \leq j \leq n$, $n = |M|$) is the set of sub-tasks of a task $A_j \in M$ (S_m, S_j is empty if respectively A_m, A_j is atomic).
$A_t = A_m$
Set *target node* = a_m (i.e. *target node* is the *initial node* of the *merge task* A_m)
Else if A is a *merge task* **then**
Let M be the associated set of *merged tasks* for A ($A \notin M$)
$S_t = S \cup S_1 \cup S_2 \ldots \cup S_n$ where S is the set of sub-tasks of A, and S_j ($1 \leq j \leq n$, $n = |M|$) is the set of sub-tasks of a task $A_j \in M$ (S, S_j is empty if respectively A, A_j is atomic)
Set *target node* = a (i.e. *target node* is the *initial node* of A)
Else
Let S be the set of sub-tasks of A
$S_t = S$
Set *target node* = a (i.e. *target node* is the *initial node* of A)
Set *target graph* = G_w where G_w is the associated workflow graph of the detected *target node*
// Step 2. Assemble evaluation set and mark processed and initial tasks.
Initialize *evaluation set* T (to be filled in the following)
For each task A_s in S_t **do**
If A_s is delegated **then**
Assemble *evaluation set* T_d from delegations by passing A_s to Algorithm 6.2
$T = T \cup T_d$
Else
$T = T \cup \{ A_s \}$
If A_s is atomic **then**
Mark A_s as *processed*
Else
Mark A_s as *initial task*
If A has an associated *merge task* A_m **then**
Let M be the associated set of *merged tasks* for A_m ($A \in M$)
Mark A_m and all tasks in M as *processed*.
Else if A is *merge task* **then**
Let M be the associated set of *merged tasks* for A ($A \notin M$)
Mark A and all tasks in M as *processed*.
Else
Mark task A as *processed*
// Step 4. Handle interpretation of hierarchical decomposition of actual task for processing.
User selects transformation of the hierarchical decomposition for the detected initial task A_t.
If A_t is transformed to sub-process **then**
Adjust the type of the *initial node* a_t for A_t in G_w to *sub-process*
Create a new workflow *graph* $G_{w.t}$ for the sub-process and set it as *target graph*
Crate a *start node* from A_t in $G_{w.t}$ and set it as *target node*
Generate a *block* from T
Insert generated *block* in $G_{w.t}$ after the *start node*
Create an *end node* and append it to the generated *block* in $G_{w.t}$
Else If A_t is transformed to an atomic task **then**
Adjust the type of the *initial node* a_t for A_t in G_w to *atomic task*
Generate *block* from T
Insert generated *block* in G_w after the a_t
Else if A_t is transformed to logical group association or omitted **then**
If A_t is transformed through merge at hierarchical level of parent task **then**
Let further T_t be the associated evaluation set of A_t, i.e. this set has been generated during the transformation of a parent task of A_t or during the first transformation of A_t

as non-atomic, delegated root task. Merge the evaluation set for the transformation of A_t with the associated evaluation set of A_t by excluding A_t itself as it is omitted or transformed to logical group association, i.e. $T = T \cup T_t \setminus \{ A_t \}$
Generate a *block* from T
Replace workflow *block* from T_t in G_w with the *block* from T
Recover logical group associations for nodes with corresponding tasks in T_t

Else if A_t is transformed without merge at hierarchical level of parent task **then**
Generate *block* from T
Replace a_t with the generated *block* from T in G_w

If A_t is transformed to logical group association **then**
Create *logical group association* L_{At} encompassing all generated workflow nodes from tasks in S_t. Each task from S_t receives a *group element association* to L_{At}. An important notice here is that in case of transformation with merge, L_{At} may encompass also workflow nodes with corresponding tasks in T_t depending on the detected sequence flow (cf. Figure 6.3).

The provided transformation algorithms consider that a workflow graph has a start node and an end node and the start node is not a gateway [VVK08, OMG06]. Start nodes in generated workflow graphs may inherit some of the attributes of the corresponding task instances from the task delegation graph. This inheritance depends on the workflow modeling notation and is discussed in the implementation in Chapter 8.

The start node and the end node are inserted when the first transformation step is performed for a workflow graph – be it in the main process or a sub-process. The idea behind that is to allow users to specify workflow graphs at different levels of detail, i.e. the user can interrupt the process model transformation by producing an underspecified but complete workflow model. The generation of a workflow block from an evaluation set is discussed in the following sections.

6.2.6 Task Processing Changes

The thesis suggests using the change history of ad-hoc task instances for the transformation of a task delegation graph to a workflow graph. The change history reflects the processing stages of tasks in the course of an ad-hoc process instance which allow estimation of temporal relationships between tasks. Different types of task changes have been discussed in Chapter 4. Task instance changes can alter the task structure, attributes, or associations to artifacts. Some of the changes that have been performed on task instances can be considered as changes that indicate processing of a given task. Such changes are referred to as *task processing changes*. Task processing changes are a specific type of task context changes (cf. Section 4.4.2), i.e. changes to task instance attributes or associations to artifacts. Task processing changes can be *explicit* or *implicit*.

Explicit task processing changes alter task attributes that explicitly denote task progress and state. Such are the *percent complete* and *status attributes*.

Implicit task processing changes on the other hand are changes to associated artifacts. Such changes can denote that a user has altered or worked on documents, which are produced or modified in a given task. Changes to task context information such as name and description can be also considered as implicit task processing changes.

As it cannot be considered that each change to an ad-hoc task instance denotes that some activity related to that task has been performed, the thesis suggests enabling end users to select which types of changes should be considered as task processing changes during process model transformation. Filtering of changes can precise the derived workflow models and can make them more consistent from business perspective.

Furthermore, it cannot be assumed that each activity on a given task has been reflected in the respective to-do item through maintaining its state, attributes, or artifacts. Thus, the resulting control flow is based on suggestions and needs to be inspected and validated explicitly by the

users after the transformation.

The time stamps of task processing changes are evaluated to determine points in time, when activities for processing a given task have been performed. These estimations result in *task ranges*, which are used to determine the control flow for derived workflow models.

6.2.7 Task Ranges

Task ranges are an approximation of the time period during which a task has been processed, i.e. during which a user has performed some activities to accomplish the task. If a task *A* has received a first task processing change at a given time t_1 and a last task processing change at given time t_n, the period t_1 to t_n is referred to as the *range* of task *A*, and t_1 and t_n are called respectively *range start time* and *range end time* for *A*. If *A* has received only one task processing change, both times overlap, i.e. the range of *A* encompasses only a single point in time $t = t_1 = t_n$.

Task ranges are a simplified way to suggest task sequencing. This approach is chosen as ad-hoc tasks can be executed without meeting any pre- or post-conditions. The resulting sequencing is hence based on suggestions and during model transformation the users are enabled to view the task change and evolution history and to estimate whether the suggested flow is correct.

Task ranges are computed for all tasks in an *evaluation set* to determine which tasks have been performed in parallel and can be transformed to parallel (*AndSplit, AndJoin*) flow, and which tasks have been performed in a strict sequence. This is the primary information that can be extracted from a task delegation graph, as task delegation graphs do not provide support for loops or definition of decision flow (*XOrSplit, XOrJoin*). Some considerations towards derivation of decision flow based on reuse of task patterns are discussed later on.

Task ranges are calculated based on the task processing changes of a given task. Example ranges are shown in Figure 6.4. If the ranges of two tasks do not overlap, the sequence of the tasks in the workflow model is determined based on the sequence of their ranges. In Figure 6.4 task A_4 is subsequent to A_5 because $t_{5.p} < t_{4.1}$, i.e. the range end time of A_5 precedes the range start time of A_4. Thus, if A_4 and A_5 are sub-tasks of the same parent task in a task delegation graph, and A_4 comes before A_5 in the hierarchy of the parent task, the resulting task sequence in the derived workflow graph will differ from the hierarchical task order in the task delegation graph.

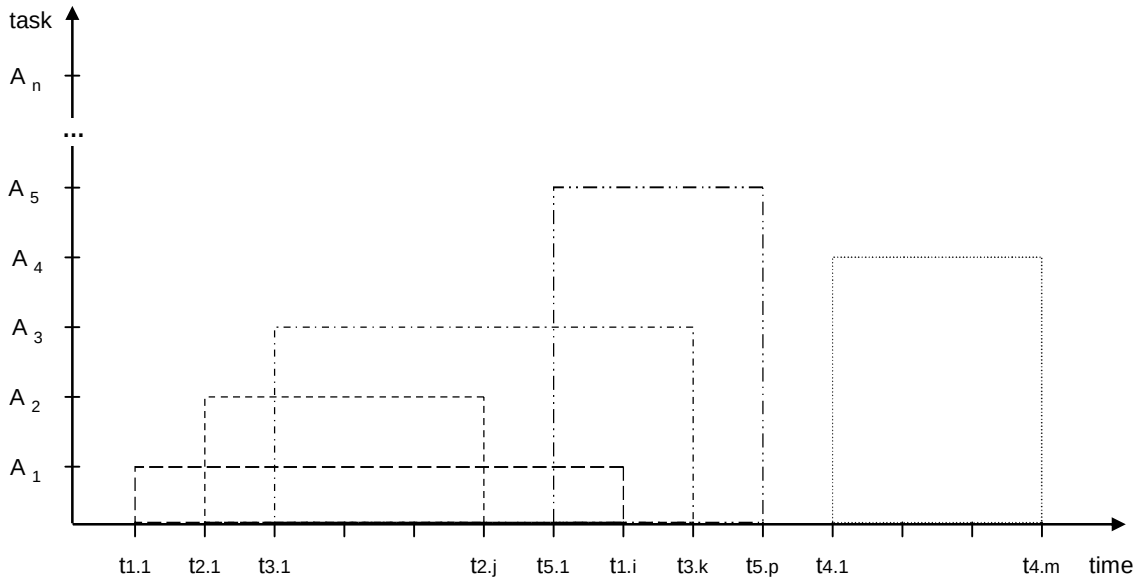


Figure 6.4: Task ranges

Overlapping ranges generally identify parallel tasks, i.e. if the range of task *A* has start and end times respectively t_1 and t_n where $t_1 \leq t_n$, each task from the evaluation set of task *A* is considered parallel to *A* if it has received a task processing change at a given time t_x such that $t_1 \leq t_x \leq t_n$. On Figure 6.4 task A_1 is parallel to tasks A_2, A_3 and A_5. Parallelism is symmetric, but not reflexive and not transitive. On Figure 6.4 task $A_2 \parallel A_3$ and $A_3 \parallel A_5$, yet $A_2 \nparallel A_5$. Overlapping ranges that produce multiple sets of parallel tasks have some complex implications with respect to the correctness criteria for derived workflow models. Different interpretations and handling of overlapping ranges are discussed in the following sections.

6.2.8 Sequence Flow and Task Ranges

A generated workflow block is a directed graph, which follows the correctness criteria for a workflow graph (cf. Section 6.2.1). The transformation of ad-hoc task structures to workflow blocks based on ad-hoc task ranges needs to consider these correctness criteria and to resolve possible inconsistencies. Such inconsistencies and the mechanisms for resolving them are discussed in the following.

6.2.8.1 Terminology

To describe the workflow block generation, the following terms are used:

- A **fork node** is a parallel split (*AndSplit*) gateway. Forks are defined for parallel tasks. A fork node can have:
  - Only one incoming edge
  - Two or more outgoing edges
- A **join node** is a parallel merge (*AndJoin*) gateway. Each fork node has an associated join node. A join node has:
  - Two or more incoming edges
  - Only one outgoing edge
- A **task node** in the following refers to any of the following: atomic task node that represents a workflow task model for an atomic task, an initial node, or a sub-process node, depending on the corresponding task from the task delegation graph in the evaluation set. All these node types are referred as task node to simplify the discussion of the workflow block generation. A task node has:
  - Only one incoming edge
  - Only one outgoing edge
- A **parallel path** is a path in a generated workflow graph from a fork node to its corresponding join node. A parallel path can contain nested forks, which together with their respective joins form nested parallel blocks in the path.

[RRD03] uses the term *"branching"* without differentiating betweens forks (*AndSplit*) and exclusive branching points (*XOrSplit*). This difference is clearly stated in [RD98, OMG06]. To avoid ambiguities, the thesis uses the terminology from [OMG06] and refers to *forking* as the dividing of a path into two or more *parallel paths*, and to *branching* as the dividing a path into two or more *alternative paths* [OMG06]. Branching is specified through *decision* nodes and respective closing *merge* nodes [OMG06].

The control flow transformation described in the following focuses on the relationships that can be derived from an ad-hoc task delegation graph. Task delegation graphs do not provide the possibility to capture alternative flows. Thus, the focus in the following is set on sequences and parallel flow.

6.2.8.2 Workflow Graph Correctness Criteria and Task Ranges

The transformation of task ranges to task sequence flow has to consider the correctness criteria for workflow graphs. The correctness criteria for a workflow graph (cf. Section 6.2.1) postulate

that control blocks (sequences, forking) can be arbitrarily nested, but they are not allowed to overlap [RD98, RRD03]. This means that there can be no edges between two parallel paths of a fork. Thus, the following holds.

Theorem 1. Let a_j and a_k be two nodes in a workflow graph such that a_k is in sequence to a_j. Let a_x be a further node in the workflow graph, which is parallel to both a_j and a_k. Then each node a_p in the workflow graph which is parallel to a_k and subsequent to a_x is also parallel to a_j.

Proof: The theorem results from the correctness criteria for a workflow graph stating that control blocks (sequence, forking, branching) cannot overlap. The initial case is shown in Figure 6.5 (a): one fork *f* with the nodes a_j and a_k in sequence, and an additional node a_x parallel to a_j and a_k in a further parallel path in *f*. The BPMN notation [OMG06] is used for all figures in the following, where forks and respective joins (parallel split and merge) are denoted with '+'. An important notice here is that there might be other nested forks, decisions and other nodes in *f*. The workflow block shown in Figure 6.5 is simplified to stress on the relationships concerning the theorem.

Let the initial case (Figure 6.5 (a)) be true, i.e. a_j and a_k are subsequent, and a_x is parallel to a_j and a_k. The conditions for a_p are manipulated to reject the corresponding opposite assumptions.

1) Let a_p be parallel to a_k (true condition from theorem). Assume that a_p is subsequent to a_j. In this case the flow given in Figure 6.5 (b) results, i.e. parallelism for a_p and a_k can be provided through a fork f_v. Thereby f_v cannot overlap with the parent fork f_u which provides parallelism for a_j and a_x, i.e. no path $P = \{f_v, \ldots, j_u, \ldots, j_v\}$ exists with j_u and j_v being the respective join nodes for f_u and f_v. As a result a_p remains parallel to a_x. This conflicts with the condition that a_p is subsequent to a_x defined in the theorem. Thus the assumption is not true, i.e. a_p cannot be subsequent to a_j.
2) Let a_p be in sequence to a_x (true condition from theorem). Assume that a_p is subsequent to a_j. In this case the flow given in Figure 6.5 (c) results, i.e. the fork *f*, which provides parallelism for a_j and a_x, is closed so that a_p can be subsequent to both - a_j and a_x. However, *f* contains a_k so that when *f* is exited, a_p remains subsequent also to a_k. This conflicts with the condition from the theorem that a_p is parallel to a_k. Thus the assumption is not true, i.e. a_p cannot be subsequent to a_j.
3) From 1) and 2) $\Rightarrow a_p \parallel a_j$. The correct workflow block is shown in Figure 6.5 (d).

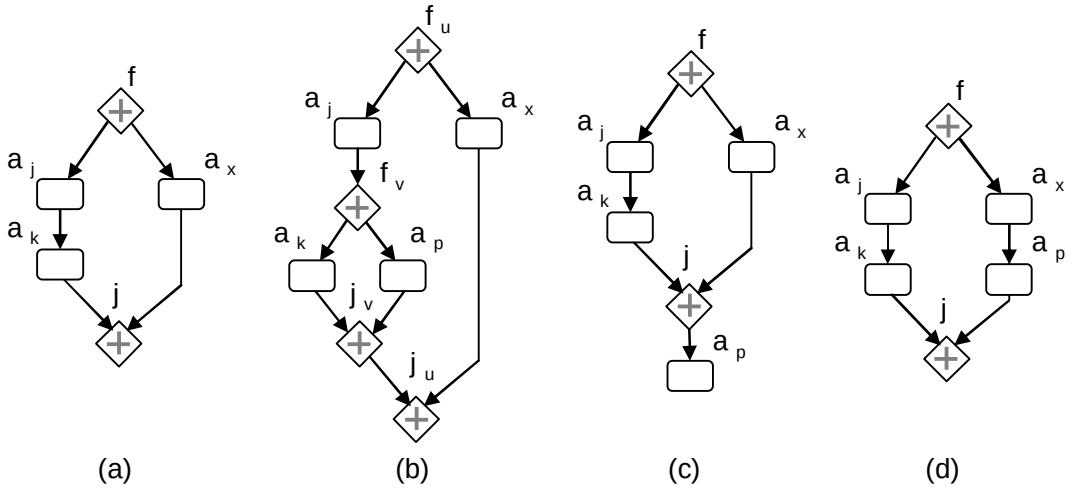


Figure 6.5: Correctness criteria for nested sequence and parallel flows

All theorems related to workflow graphs in the thesis apply for both forks and decisions as both gateway types have different semantics at runtime but require the same structural relationships at design time. For simplicity, only forks are discussed in the thesis. Further, the theorems apply for all kinds of nodes in a workflow graph. Task nodes are shown in the figures for simplicity.

Theorem 1 has important implications for the transformation of task ranges to task sequence flow. To clarify these implications, the following characteristic of task ranges is considered.

Theorem 2. Let A_j and A_k be two tasks from a task delegation graph, such that their ranges do not overlap and the range of A_k is subsequent to the range of A_j. Let A_x be a task from the task delegation graph, the range of which overlaps with the ranges of both tasks A_j and A_k. Then the range of each task A_p which is subsequent to the range of A_x, is subsequent also to the range of A_j. In other words, there exists no such A_p, that is subsequent to A_x and parallel to A_j.

Proof: Let t_{sAj} and t_{eAj} be respectively the range start time and range end time of A_j, t_{sAk} and t_{eAk} the range start time and range end time of A_k etc. The relationships concerning the theorem are shown in Figure 6.6.

1) Per definition, $t_{sAy} \leq t_{eAy}$ for all range start and end times t_{sAy} and t_{eAy} with $y \in \{j, x, k, p\}$
2) If the ranges of A_j and A_k are in sequence, then $t_{sAj} \leq t_{eAj} < t_{sAk} \leq t_{eAk}$
3) If the range of A_x overlaps with the range of A_k then the following alternatives exist:

3.1) $t_{sAk} \leq t_{sAx} \leq t_{eAk}$ In this case from 1) $t_{sAx} \leq t_{eAx} \Rightarrow t_{sAk} \leq t_{eAx}$.

3.2) $t_{sAk} \leq t_{eAx} \leq t_{eAk}$

3.3) $t_{sAx} \leq t_{sAk} \leq t_{eAx}$

3.4) $t_{sAx} \leq t_{eAk} \leq t_{eAx}$ In this case from 1) $t_{sAk} \leq t_{eAk} \Rightarrow t_{sAk} \leq t_{eAx}$.

4) From 2) and 3) $\Rightarrow t_{eAj} < t_{sAk} \leq t_{eAx}$
5) If the ranges of A_x and A_p are in sequence, then $t_{sAx} \leq t_{eAx} < t_{sAp}$
6) From 4) and 5) $\Rightarrow t_{eAj} < t_{sAk} \leq t_{eAx} < t_{sAp}$, i.e. $t_{eAj} < t_{sAp}$. Thus the range of task A_p is always subsequent to the range of task A_j.

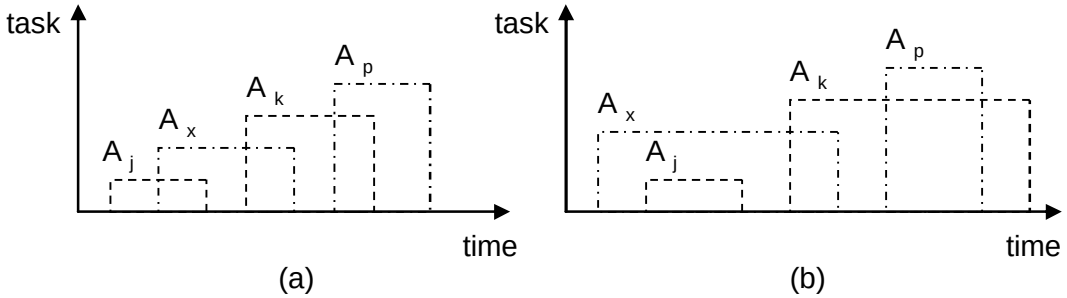


Figure 6.6: Example overlapping sequence and parallel ranges

If A_p is parallel to A_k, a conflict arises between task ranges and workflow graph correctness criteria according to Theorem 2 and Theorem 1. For dealing with such conflicts the thesis suggests a set of *consolidation* mechanisms. To clarify the consolidation, further characteristics of workflow graphs and task ranges are introduced in the following.

Theorem 3. Let a_j and a_x be two nodes in a workflow graph such that a_j is parallel to a_x. Then each task a_k that is in sequence to a_j is either parallel to a_x or in sequence to a_x.

Proof: The theorem results from the correctness criteria for a workflow graph stating that control blocks (sequence, forking, branching) cannot overlap. Let a_j and a_x be parallel. The parallelism is

provided through a fork *f* as shown in Figure 6.7 (a). In the overall workflow graph, a_k can be within or outside *f*.

1) Let a_k reside in a parallel path in *f* (see Figure 6.7 (b)). In this case, a_k is parallel to a_x.
2) Let a_k reside outside *f* (see Figure 6.7 (c)). In this case, a_k is subsequent also to a_x.

Theorem 4. Let a_j and a_x be two nodes in a workflow graph such that a_j is parallel to a_x. Then each task a_k, to which a_x is in sequence, is either parallel to a_j or precedes a_j in a strict sequence.

Proof: The theorem results from the correctness criteria for a workflow graph stating that control blocks (sequence, forking, branching) cannot overlap. Let a_j and a_x be parallel. The parallelism is provided through a fork *f* as shown in Figure 6.7 (a). In the overall workflow graph, a_k can be within or outside *f*.

1) Let a_k reside in a parallel path in *f* (see Figure 6.7 (d)). In this case a_k is parallel to a_j.
2) Let a_k reside outside *f* (see Figure 6.7 (e)). In this case, a_j is subsequent to a_k.

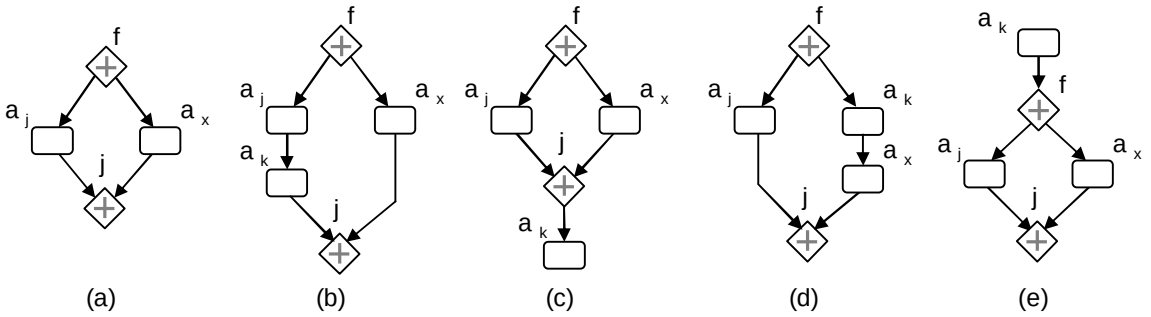


Figure 6.7: Correctness criteria for two parallel tasks and a third sequential task

The following characteristics of task ranges ensure compliance of task ranges with the workflow graph correctness criteria according to Theorem 3 and Theorem 4 (cf. Figure 6.8).

Theorem 5. Let A_j and A_x be two tasks from a task delegation graph, such that the ranges of A_j and A_x overlap. Then the range of each task A_k that is subsequent to the range of A_j either overlaps with the range of A_x or is subsequent to the range of A_x.

Proof: Let t_{sAj} and t_{eAj} be respectively the range start time and range end time of A_j, t_{sAx} and t_{eAx} the range start time and range end time of A_x, and t_{sAk} and t_{eAk} the range start time and range end time of A_k.

1) Per definition, $t_{sAy} \leq t_{eAy}$ for all range start and end times t_{sAy} and t_{eAy} with $y \in \{j, x, k\}$.
2) Assume that:

2.1) The range of A_k does not overlap with the range of A_x: $t_{sAk} \leq t_{eAk} < t_{sAx} \leq t_{eAx} \vee t_{sAx} \leq t_{eAx} < t_{sAk} \leq t_{eAk}$

and also

2.2) The range of A_k is not subsequent to the range of A_x: $\overline{t_{sAx} \leq t_{eAx} < t_{sAk} \leq t_{eAk}}$

2.3) From 2.1) and 2.2) $\Rightarrow t_{sAk} \leq t_{eAk} < t_{sAx} \leq t_{eAx}$

Reject the assumption:

3) If the ranges of A_j and A_k are in sequence, then $t_{sAj} \leq t_{eAj} < t_{sAk} \leq t_{eAk}$
4) If the range of A_j overlaps with the range of A_x then the following alternatives exist:

4.1) $t_{sAj} \leq t_{sAx} \leq t_{eAj}$ In this case from 3) $t_{eAj} < t_{sAk} \Rightarrow t_{sAx} < t_{sAk}$ conflicts with 2.3)

4.2) $t_{sAj} \leq t_{eAx} \leq t_{eAj}$ In this case from 3) $t_{eAj} < t_{sAk} \Rightarrow t_{eAx} < t_{sAk}$ conflicts with 2.3)

4.3) $t_{sAx} \le t_{sAj} \le t_{eAx}$ In this case from 3) $t_{sAj} \le t_{eAj} < t_{sAk} \Rightarrow t_{sAx} < t_{sAk}$ conflicts with 2.3)
4.4) $t_{sAx} \le t_{eAj} \le t_{eAx}$ In this case from 3) $t_{sAj} \le t_{eAj} < t_{sAk} \Rightarrow t_{sAx} < t_{sAk}$ conflicts with 2.3)
5) From 4) $\Rightarrow$ the assumption is wrong, i.e. the range of A_k either overlaps with the range of A_x or is subsequent to it.

Theorem 5 complies with Theorem 3. Figure 6.8 (a) shows an example of tasks A_j and A_x with overlapping ranges and a further task A_k the range of which overlaps with that of A_x. Figure 6.8 (b) shows an example case when the range of A_k is in sequence to the range of A_x.

Theorem 6. Let A_j and A_x be two tasks from a task delegation graph, such that the ranges of A_j and A_x overlap. Then the range of each task A_k, to which the range of A_x is subsequent, either overlaps with the range of A_j or the range of A_j is in sequence to the range of A_k.

Proof: Let t_{sAj} and t_{eAj} be respectively the range start time and range end time of A_j, t_{sAx} and t_{eAx} the range start time and range end time of A_x, and t_{sAk} and t_{eAk} the range start time and range end time of A_k.

1) Per definition, $t_{sAy} \le t_{eAy}$ for all range start and end times t_{sAy} and t_{eAy} with $y \in \{j, x, k\}$.
2) Assume that:
2.1) The range of A_k does not overlap with the range of A_j: $t_{sAj} \le t_{eAj} < t_{sAk} \le t_{eAk} \vee t_{sAk} \le t_{eAk} < t_{sAj} \le t_{eAj}$
and also
2.2) The range of A_j is not subsequent to the range of A_k: $\overline{t_{sAk} \le t_{eAk} < t_{sAj} \le t_{eAj}}$
2.3) From 2.1) and 2.2) $\Rightarrow t_{sAj} \le t_{eAj} < t_{sAk} \le t_{eAk}$
Reject the assumption:
3) If the ranges of A_k and A_x are in sequence, then $t_{sAk} \le t_{eAk} < t_{sAx} \le t_{eAx}$
4) If the range of A_j overlaps with the range of A_x then the following alternatives exist:
4.1) $t_{sAj} \le t_{sAx} \le t_{eAj}$ In this case from 3) $t_{eAk} < t_{sAx} \Rightarrow t_{eAk} < t_{eAj}$ conflicts with 2.3)
4.2) $t_{sAj} \le t_{eAx} \le t_{eAj}$ In this case from 3) $t_{eAk} < t_{sAx} \le t_{eAx} \Rightarrow t_{eAk} < t_{eAj}$ conflicts with 2.3)
4.3) $t_{sAx} \le t_{sAj} \le t_{eAx}$ In this case from 3) $t_{eAk} < t_{sAx} \Rightarrow t_{eAk} < t_{sAj}$ conflicts with 2.3)
4.5) $t_{sAx} \le t_{eAj} \le t_{eAx}$ In this case from 3) $t_{eAk} < t_{sAx} \Rightarrow t_{eAk} < t_{eAj}$ conflicts with 2.3)
5) From 4) $\Rightarrow$ the assumption is wrong, i.e. the range of A_k either overlaps with the range of A_j or the range A_j is in sequence to the range of A_k.

Theorem 6 complies with Theorem 4. Figure 6.8 (c) shows an example of tasks A_j and A_x with overlapping ranges and a further task A_k the range of which precedes that of A_x in a strict sequence, and overlaps with the range of A_j. Figure 6.8 (d) shows an example where the range of A_k precedes the ranges of both A_x and A_j in a strict sequence. The discussed theoretical foundations are used to *consolidate* tasks according to the workflow graph correctness criteria during the generation of a workflow block as discussed in the next sections.

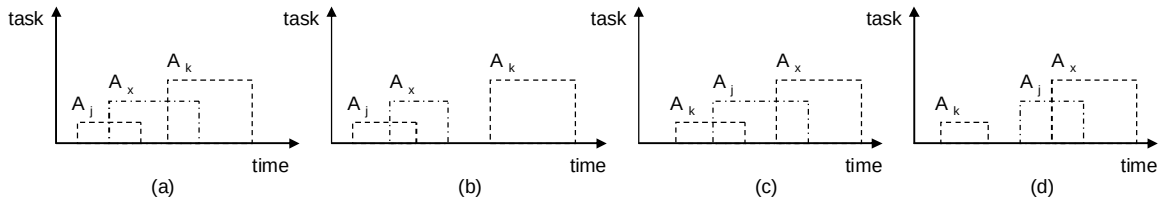


Figure 6.8: Example ranges for two parallel tasks and a third sequential task

6.2.9 Sequence Flow Generation

The sequence flow is derived from an evaluation set through the following basic operations:

1) *Compute ranges*
2) *Sort the evaluation set*
3) *Assemble parallel sets*
4) *Consolidate parallel sets for sequence flow generation*
5) *Generate sequence flow*

The **compute ranges** operation calculates the task ranges based on the timestamps of the first and last processing changes for all tasks in the evaluation set.

The **sort evaluation set** operation can be performed in a single iteration over the evaluation set with the computing of the task ranges. Sorting produces an ordered evaluation set, where tasks are ordered based on their ranges starting with the task that has the first range start time. For example, the ordered evaluation set for the tasks in Figure 6.6 (a) is $O = \{A_j, A_x, A_k, A_p\}$, and for Figure 6.6 (b) $O = \{A_x, A_j, A_k, A_p\}$. Thus task sequence in the evaluation set reflects the sequence that is detected based on the task ranges. If two tasks have ranges with the same start time, the tasks are ordered in a sequence (it is not important which of them comes first).

Having an ordered evaluation set is important for producing valid sequence flow through the algorithms described later on. In the following a task A_k from an ordered evaluation set is said to be *subsequent* to another task A_j from the evaluation set, if the index of A_k in the evaluation set is higher than the index of A_j.

The **assemble parallel sets** operation iterates over the ordered evaluation set and uses the computed ranges to detect which tasks have overlapping ranges based on the range start and end times. If the ranges of two tasks A_j and A_x from the ordered evaluation set overlap, A_j is added to a *parallel set* of A_x, and A_x is added to a parallel set of A_j (symmetric behavior). A task A_j from an ordered evaluation set is said to be *parallel* to another task A_x from the evaluation set, if the parallel set of A_j contains A_x and vice versa.

The **consolidate parallel sets** operation resolves discrepancies in overlapping ranges, which conflict with the correctness criteria for workflow graphs as discussed in Section 6.2.8.2. Consolidation is discussed in details in the next section.

The **generate sequence flow** operation uses the ordered evaluation set and the consolidated *parallel sets* to generate a workflow block as described later on.

6.2.9.1 Consolidation of Inconsistent Task Ranges

A set of consolidation options is considered to correct inconsistent task ranges according to the workflow graph correctness criteria. The described transformation method generally preserves all *sequence* relationships from the ordered evaluation set and allows manipulating the *parallel* relationships between tasks to resolve inconsistent ranges. Consolidation is performed before the workflow block generation, so that during that generation all relationships in the parallel sets are consistent. Consolidation runs iteratively over the evaluation set until no more inconsistent tasks occur. Consolidation is performed for each set of tasks A_j, A_k, A_x, and A_p in the evaluation set for which the following is true (cf. also Theorem 1 and Theorem 2):

- A_k is *subsequent* to A_j
- A_x is *parallel* to both A_j and A_k
- A_p is *subsequent* to both A_j and A_x, and *parallel* to A_k

Example consolidations for task ranges are given in Figure 6.9, where (a) gives the inconsistent task ranges and the (b) to (g) show examples of a generated workflow block for each consolidation, provided that the evaluation set contains only the given four tasks. Implications from multiple nested ranges are discussed in the following sections for the different consolidation options. Additional consolidations are performed when workflow models are redesigned based on ad-hoc deviation tasks from multiple workflow instances as discussed in Section 6.2.12.4 later on.

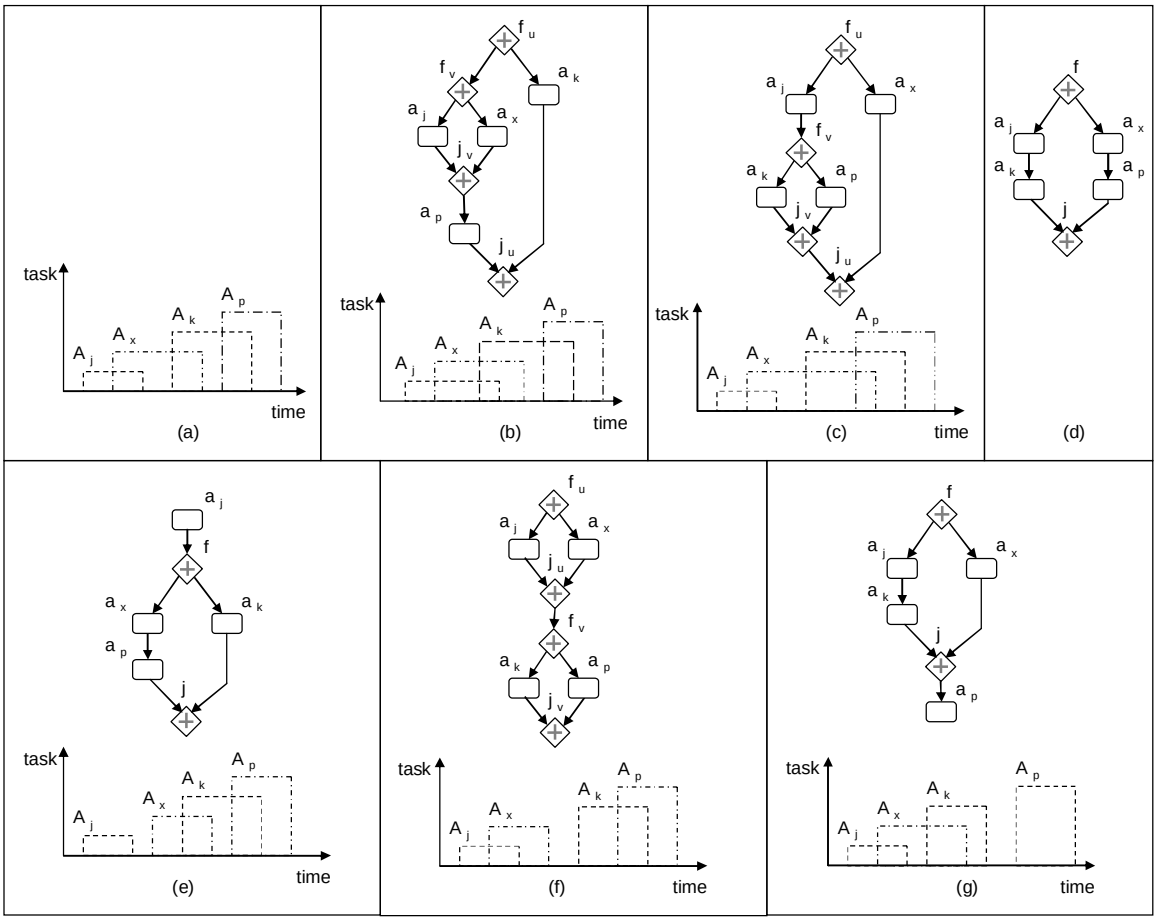


Figure 6.9: Example consolidations

6.2.9.1.1 Merge Consolidations – Extending Parallel Relationships

Three types of consolidation for extending parallel relationships are considered in the thesis:

- **Merge head** consolidation adds parallelism between A_j and A_k. An example consolidation of this type is shown in Figure 6.9 (b).
- **Merge tail** consolidation adds parallelism between A_x and A_p. An example consolidation of this type is shown in Figure 6.9 (c).
- **Merge tail to head** consolidation adds parallelism between A_j and A_p. An example consolidation of this type is shown in Figure 6.9 (d). This consolidation type has no equivalent representation through task ranges (cf. Theorem 2). It is intended for consistency according to Theorem 1.

For all *merge* consolidations the following applies. Let A_1 and A_2 be the two tasks between which parallelism is added in a merge consolidation such that A_2 is subsequent to A_1 in the evaluation set, e.g. in case of merge head consolidation $A_1 = A_j$ and $A_2 = A_k$, in case of merge tail consolidation $A_1 = A_x$ and $A_2 = A_p$ etc. In order to keep the workflow graph correctness criteria for all merge consolidations it is intrinsic to add parallelism for all tasks A_y that are subsequent to A_1 and to which A_2 is subsequent in the evaluation set. Concretely, according to Theorem 3, if A_1 is parallel to A_2, each task A_y that is subsequent to A_1 needs to be either parallel or subsequent to A_2. On the other hand, according to Theorem 4, if A_1 is parallel to A_2, each task to which A_2 is subsequent, needs to be either parallel to A_1 or to precede A_1 in a strict sequence. To preserve a

given sequence for A_y when parallelism between A_1 and A_2 is added, a *direction* for adding of parallelism between A_1 and A_2 is additionally considered. The direction can be: (i) *to tail* and (ii) *to head*. The *to tail* direction adds parallelism between A_1 and each A_y, thus preserving the sequence between A_y and A_2 in compliance with Theorem 4, i.e. A_1 is parallel to A_2, A_y precedes A_2 in strict sequence and is parallel to A_1. The *to head* direction adds parallelism between each A_y and A_2, thus preserving the sequence between A_1 and A_y in compliance with Theorem 3, i.e. A_1 is parallel to A_2, A_y is subsequent to A_1 and parallel to A_2. The merge procedure is given in Algorithm 6.6 later on. The thesis considers each task A_y which parallel relationships need to be additionally altered during a given consolidation as a *nested* task for the consolidation.

Merge consolidations deliver an optimized workflow model with respect to *parallelism* [Rei05]. However the model needs to be checked for consistency from business perspective as some of the merged tasks may not be appropriate for parallel execution. For example, if the input for one task is delivered as output from another task, merging both tasks as parallel is wrong and would cause inconsistencies when executing the workflow.

6.2.9.1.2 Split Consolidations – Removing Parallel Relationships

As an alternative to the merge consolidations the following three types of split consolidations are proposed, which remove parallel relationships between tasks:

- **Split head** consolidation removes parallelism between A_j and A_x. An example consolidation of this type is shown in Figure 6.9 (e).
- **Split middle** consolidation removes parallelism between A_x and A_k. An example consolidation of this type is shown in Figure 6.9 (f).
- **Split tail** consolidation removes parallelism between A_k and A_p. An example consolidation of this type is shown in Figure 6.9 (g).

For all *split* consolidations the following applies. Let A_1 and A_2 be the two tasks between which parallelism is removed in a split consolidation such that A_2 is subsequent to A_1 in the evaluation set, e.g. in Figure 6.9 in case of split head consolidation $A_1 = A_j$ and $A_2 = A_x$, in case of split middle consolidation $A_1 = A_x$ and $A_2 = A_k$ etc. In order to keep the workflow graph correctness criteria during split it is intrinsic to remove parallelism between A_1 and all nested tasks A_y that are parallel to A_1 and subsequent to A_2, and additionally to remove parallelism between A_2 and all nested tasks A_z that are parallel to A_2 and to which A_1 is subsequent in the evaluation set. For example, during split head consolidation of evaluation sets where A_x comes before A_j, e.g. resulting from inconsistent ranges as shown in Figure 6.6 (b), parallelism needs to be removed between A_x and each nested task A_y from the evaluation set that is parallel to A_x but subsequent to A_j. If this parallelism is not removed, after the consolidation A_x will remain parallel to A_y, and additionally A_j will be in sequence to A_x, and A_y will be in sequence to A_j. According to Theorem 3, if A_x and A_y are parallel, and A_j is subsequent to A_x, then A_j can be either parallel to A_y or subsequent to A_y. Thus to preserve the current sequence of A_j and A_y when the consolidation removes parallelism between A_j an A_x, the parallelism between A_x and A_y needs to be also removed. Similarly during split tail consolidation for evaluation sets resulting from inconsistent ranges as shown in Figure 6.6 (b), it is intrinsic to remove also parallelism between A_k and each task A_y from the evaluation set, that is parallel to A_k but subsequent to A_p. The removal of parallelism is performed recursively as given in Algorithm 6.6 later on.

Split consolidations deliver a workflow model, which may replace potential parallel flows with strict sequences. Thus the delivered workflow model is not fully optimized with respect to parallel flow, which can cause unnecessary *"wait times"* [Rei05] during workflow execution. However, the split options keep the process consistent in that no false parallel flows are introduced. The resulting workflow model can be thus checked for optimization possibilities.

6.2.9.1.3 Consolidation Support

All consolidation options consider altering a single temporal relationship between two tasks to minimize the differences between the derived workflow model and the captured, real-life ad-hoc process execution. However, when there are multiple, overlapping sets of inconsistent ranges, the consolidation is performed recursively and can affect multiple tasks. Thereby different consolidation options lead to completely different workflow models. Therefore, transformations where consolidation has been performed require explicit check of the derived workflow model to ensure its consistency. To facilitate the early process model validation, the thesis suggests that consolidation should be supported through a visual environment, where the user who performs the transformation is enabled to see the consolidated tasks, to evaluate their change history and original task delegation graph. Through this, the user can manually select the consolidation options that they consider as delivering the highest consistency for the derived process model. Thereby the user is required to *specify* the temporal relationships between inconsistent tasks rather than to model them from scratch. The thesis further suggests supporting user-defined consolidation through history of the performed consolidations. During consolidation, this history can help the user to avoid performing controversial steps, e.g. removing and adding parallelism between the same tasks in subsequent consolidation steps. After the workflow model derivation is complete, the consolidation history can be used to elaborate on the performed consolidations and to check if the generated sequence flow is appropriate and to adapt it manually if needed. Validation of consolidations can be further supported through tagging or visual marking of consolidated tasks, so that the user is enabled to identify these tasks in the derived workflow model and to view their consolidation and change history.

As an alternative to the manual consolidation it is further possible to perform a preliminary assessment of the effects of different consolidation options and to compute an optimal set of consolidations for a given evaluation set. As such optimal consolidation set can be considered the combination of consolidations that alters minimal number of relationships between tasks in the evaluation set. This optimal consolidation set can be used to automatically derive a workflow model that is as close as possible to the captured ad-hoc task change history. However, such automated transformation does not allow the user to perform an early validation of inconsistent task ranges. Thus an increased effort may be required for manual validation and adaptation of the derived workflow model later on. Hence, automated consolidation is not considered in the thesis.

6.2.9.1.4 Consolidation Algorithms

This section provides the consolidation algorithms. Algorithm 6.5 specifies the overall iteration over the evaluation set for detecting inconsistent task relationships. The merge and split consolidations are performed respectively through Algorithm 6.6 and Algorithm 6.7.

Algorithm 6.5: Consolidation of parallel sets for inconsistent task relationships.

Require: *evaluationSet* is an ordered evaluation set for workflow block generation.

```
While true
    For j = 0 to evaluationSet.count – 1 do
        For k = j + 1 to evaluationSet.count – 1 do
            taskA = evaluationSet.item(j)
            taskB = evaluationSet.item(k)
            If not taskB.parallelSet.contains(taskA) then
                // taskB is in sequence to taskA
                For each taskC in taskB.parallelSet do
                    If taskA.parallelSet.contains(taskC) then
```

// taskC is parallel to both taskA and taskB
For each taskD in taskB.parallelSet **do**
If taskD is taskC **then**
Continue for
If not taskD.parallelSet.contains(taskC)
And also not taskA.parallelSet.contains(taskD) **then**
Show the consolidation environment. taskA, taskB, taskC, taskD, and the evaluation set are used as input to detect all tasks that are affected by a selected consolidation. Detection is performed in the same manner as during the actual consolidation given below. The user is enabled to choose a consolidation option by viewing all relationships that will be changed by a given consolidation, and the consolidation and task change histories. For merge operations a *mergeToTail* flag is set, which indicatines the merge direction. When the user confirms a given consolidation, a consolidation entity is created for each two tasks, which relationships are altered. The entity specifies the change of the tasks' relationships (parallelism added or removed). It is associated to these tasks and added to the consoliation history. The tasks themselves are tagged as consolidated task. The actual consolidation is performed by calling Algorithm 6.6 and Algorithm 6.7 respectively for merge and split consolidations with the appropriate input parameters as given in the following.
If merge_head **then**
merge(taskA, taskB, evaluationSet, mergeToTail)
Continue while
Else if merge_tail **then**
merge(taskC, taskD, evaluationSet, mergeToTail)
Continue while
Else if merge_tail_to_head **then**
merge(taskA, taskD, evaluationSet, mergeToTail)
Continue while
Else if split_head **Then**
split(taskA, taskC, evaluationSet, null)
Continue while
Else if split_middle **then**
split(taskC, taskB, evaluationSet, null)
Continue while
Else if split_tail **then**
split(taskB, taskD, evaluationSet, null)
Continue while
Else if ad-hoc deviation tasks from multiple workflow instances are contained in the evaluationSet **then**
// taskB is parallel to taskA and additionally ad-hoc deviation tasks from
// different workflow instances are transformed, which require further
// consolidations as discussed in Section 6.2.12.4.
For m = j + 1 to k **do**
// Check for a taskE, which is subsequent to taskA and to which taskB is
// subsequent. As taskA and taskB are parallel, the existance of a taskE

```
                    // causes inconsistency according to Theorem 3 and Theorem 4 and
                    // requires additional consolidation.
                    taskE = evaluationSet.item(m)
                    If not taskA.parallelSet.contains(taskE)
                    and also not taskB.parallelSet.contains(taskE) then
                        Show the consolidation environment. taskA, taskB, taskE, and the
                        evaluation set are used as input to detect all tasks that are affected by
                        a selected consolidation. Detection is performed in the same manner
                        as during the actual consolidation given below. The user is enabled
                        to choose a consolidation option by viewing all relationships that will
                        be changed by a given consolidation, a mergeToTail flag is set for
                        merge operations, and a consolidation entity is managed analogously
                        to the common consolidation discussed above.
                        If merge_head then
                            merge(taskA, taskE, evaluationSet, mergeToTail)
                            Continue while
                        Else if merge_tail then
                            merge(taskE, taskB, evaluationSet, mergeToTail)
                            Continue while
                        Else if split then
                            split(taskA, taskB, evaluationSet, null)
                            Continue while
            // Exit the outer while loop if no consolidations have been performed.
            Exit while
```

Algorithm 6.6: Merge consolidation.

Require:

1) *task1* and *task2* are two subsequent tasks for which parallelism should be added.
2) *evaluationSet* is an ordered evaluation set for workflow block generation.
3) *mergeToTail* is a boolean flag, indicating the merge direction.

```
//Add parallelism between the given tasks.
task1.parallelSet.add(task2)
task2.parallelSet.add(task1)
// Add parallelism for all affected tasks by considering the merge direction.
index1 = evaluationSet.indexOf(task1)
index2 = evaluationSet.IndexOf(task2)
For i = index1 + 1 to index2 – 1 do
    taskInBetween = evaluationSet.item(i)
    If not task1.parallelSet.contains(taskInBetween)
    and also not task2.parallelSet.contains(taskInBetween) then
        If mergeToTail then
            task1.parallelSet.add(taskInBetween)
            taskInBetween.parallelSet.add(task1)
        Else
            task2.parallelSet.add(taskInBetween)
            taskInBetween.parallelSet.add(task2)
```

Algorithm 6.7: Split consolidation.

Require:

1) *task1* and *task2* are two parallel tasks that should be split.
2) *evaluationSet* is an ordered evaluation set for workflow block generation.
3) *handledTasksMap* is a map that is used to cache already handled tasks in order to avoid handling of already split tasks. The map assigns to a given task a set of tasks which have been removed from the parallelSet of that task.

```
// Initialize a map for caching to avoid repeated handling of already split tasks.
If handledTasksMap is null then
    Initialize handledTasksMap
Else if handledTasksMap.contains(task1.identifier)
and also ((Set)handledTasksMap.item(task1.identifier)).contains(task2)) then
    Exit algorithm
// Add both task1 and task2 to the handledTasksMap.
If not handledTasksMap.containsKey(task1.identifier) then
    // Initialize a set for storing all tasks that have been removed from the parallelSet of task1.
    Initialize splitTasksSet
    splitTasksSet.add(task2)
    // Add the list to the map for task1 – map.add(value, key).
    handledTasksMap.add(splitTasksSet, task1.identifier)
Else
    // A list with split tasks for task1 already exists – get the list through map.item(key).
    splitTasksSet = handledTasksMap.item(task1.identifier)
    If not splitTasksSet.contains(task2) then
        splitTasksSet.add(task2)
// Add task2 to the handledTasksMap in the same way as task1 (see above).
If not handledTasksMap.containsKey(task2.identifier) then
    Initialize splitTasksSet
    splitTasksSet.add(task1)
    handledTasksMap.add(splitTasksSet, task2.identifier)
Else
    splitTasksSet = handledTasksMap.item(task2.identifier)
    If not splitTasksSet.contains(task1) then
        splitTasksSet.add(task1)
//Remove parallelism between the given tasks.
task1.parallelSet.remove(task2)
task2.parallelSet.remove(task1)
// Determine the task sequence resulting from the split.
index1 = evaluationSet.indexOf(task1)
index2 = evaluationSet.indexOf(task2)
firstTask = task1
firstTaskIdx = index1
secondTask = task2
secondTaskIdx = index2
If index2 < index1 then
    firstTask = task2
    firstTaskIdx = index2
    secondTask = task1
    secondTaskIdx = index1
```

```
// Initialize sets with tasks that need to be removed from the first and the second task
Initialize parallelTasksForRemoveFromFirstTask
Initialize parallelTasksForRemoveFromSecondTask
// Check for parallelism between firstTask and tasks that are subsequent to secondTask.
For each parallelTask in firstTask.parallelSet do
    index3 = evaluationSet.indexOf(parallelTask)
    If index3 > secondTaskIdx
    and also not secondTask.parallelSet.contains(parallelTask) then
        parallelTasksForRemoveFromFirstTask.add(parallelTask)
// Check for parallelism between secondTask and tasks to which firstTask is subsequent.
For each parallelTask in secondTask.parallelSet do
    index3 = evaluationSet.indexOf(parallelTask)
    If index3 < firstTaskIdx
    and also not firstTask.parallelSet.contains(parallelTask) then
        parallelTasksForRemoveFromSecondTask.Add(parallelTask)
// Call Algorithm 6.6 recursively to remove parallelism between all affected tasks.
For each parallelTask in parallelTasksForRemoveFromFirstTask do
    split(firstTask, parallelTask, evaluationSet, handledTasksMap)
For each parallelTask in parallelTasksForRemoveFromSecondTask do
    split(parallelTask, secondTask, evaluationSet, handledTasksMap)
```

6.2.9.2 Workflow Block Generation

This section describes the algorithm for workflow block generation from an ordered evaluation set and the consolidated parallel sets of the contained task instances. taskX refers to an ad-hoc task instance from a task delegation graph, and nodeX refers to the corresponding derived workflow task node. taskX.node denotes the workflow task node for taskX, and nodeX.task denotes the ad-hoc task (origin) for nodeX in a task delegation graph (cf. Figure 4.2).

Algorithm 6.8: Workflow block generation from an ordered evaluation set.

Require:

1) *evaluationSet* is an ordered evaluation set for workflow block generation.
2) *targetNode* is a node in the workflow graph at which the generated block should be appended – i.e. this is the *target node* detected in Algorithm 6.3 and Algorithm 6.4.

```
// Initialize a variable for holding a reference to the node in the generated workflow block,
// after which the node for the currently handled task will be inserted.
Initialize previousNodeInBlock
// Initialize a variable lastCreatedNode for holding a reference to the node which was created
// in the previous iteration step for processing the evaluation set.
Initialize lastCreatedNode = targetNode
For each taskA in evaluationSet do
    // Check if a workflow node has already been created for taskA.
    If not taskA.node is null then
        Continue for
    // Find the node after which the block for taskA should be inserted. Therefore, initialize
    // an ordered set for tasks to which the currently handled task is in sequence.
    Initialize preSequenceTasksSet
```

Block 1

```
For each taskB in taskA.parallelSet do
    // Look for a task from the parallel set of taskA, for which a node already exists, i.e.
    // which is parallel also to another task instance that is handled in a previous iteration.
    If not taskB.node is null then
        For each taskC in taskB.parallelSet do
            // Check only tasks that are before taskA in the evaluation set.
            If taskC is taskA then
                Exit For
            If not taskC.parallelSet.contains(taskA) then
                // taskA is parallel to taskB, and taskC is parallel to taskB, but taskA is
                // not parallel to taskC. As the parallel sets are ordered and taskB is
                // already handled, then there is a nodeC resulting from the processing of
                // taskB, and the generated nodeA is subsequent to nodeC. For example,
                // referring to Figure 6.8 (a) taskA = A_k, taskC = A_j and taskB = A_x and
                // the corresponding control flow is as shown in Figure 6.5 (a), with
                // nodeA = a_k, nodeC = a_j and nodeB = a_x.
                preSequenceTasksSet.add(taskC)
If preSequenceTasksSet.count > 0 then
    // Initialize taskD as the last task from the evaluation set to which taskA is sequential.
    Initialize taskD = preSequenceTasksSet.item(preSequenceTasksSet.count – 1)
    // If only one pre-sequence task is detected, nodeA should be inserted after the
    // workflow node of that pre-sequence task.
    previousNodeInBlock = taskD.node
    // Check if pre-sequence tasks are in nested fork(s) to add nodeA after the respective
    // join node(s). A fork exists always for two or more parallel tasks.
    If preSequenceTasksSet.count > 1 then
        Initialize preSeqParentFork = taskD.node.parentFork
        Initialize previousFork = null
        While not preSeqParentFork is null do
            // Check if preSeqParentFork contains any of the parallel tasks of taskA. In
            // this case the fork would contain also taskA. Thus previousNodeInBlock
            // remains the lastPreSeqTask or the join node of a previously detected fork.
            // In the following a node n is considered as being in a fork f and a fork f is
            // considered as containing a node n if a recursive call to the parentFork
            // relationship on n returns f, e.g. n.parentFork.parentFork.parentFork = f
            If preSeqParentFork contains any of the tasks in taskA.parallelSet then
                If not previousFork is null then
                    previousNodeInBlock = previousFork.joinNode
                Exit while
            Else if all tasks from the preSequenceTasksSet that have an associated
            workflow node are in preSeqParentFork then
                // If the preSeqParentFork does not contain a parallel task of taskA and it
                // contains all pre-sequence tasks of taskA, select the join of that fork as
                // previous node.
                previousNodeInBlock = preSeqParentFork.joinNode
                Exit while
            previousFork = preSeqParentFork
            preSeqParentFork = preSeqParentFork.parentFork
```

```
If previousNodeInBlock is null then
Block 2 {
    // taskA is not parallel to an already handled node. So, find and exit the top-most fork
    // containing lastCreatedNode.
    Initialize lastParentFork = lastCreatedNode.parentFork
    While not lastParentFork is null do
        previousNodeInBlock = lastParentFork.joinNode
        lastParentFork = lastParentFork.parentFork
}
If previousNodeInBlock is null then
    // If lastCreatedNode exists and is not in a fork then the block for taskA will be in
    // sequence to it. An important notice here is that if lastCreatedNode is passed to
    // the algorithm before the first iteration over the evaluation set, the generated block
    // will be inserted after the given lastCreatedNode in the workflow graph.
    previousNodeInBlock = lastCreatedNode
// Initialize an ordered set where all tasks that should be added in a fork will be inserted.
Initialize a set parallelSetForFork
// Initialize an ordered set, containing ordered sets of tasks that are parallel among each
// other but to which a further task that is parallel to taskA is subsequent. The contained
// ordered sets are used to generate nested forks as discussed in the following.
Initialize nestedForksSet
// Check if a fork needs to be added
If taskA.parallelSet.count > 0
and also not all tasks in taskA.parallelSet have already assigned workflow nodes then
    // Initialize a set where all tasks that need to be excluded from the fork of taskA will
    // be temporarily stored.
    Initialize a set excludeFromFork
    // Initialize a (hash) map, which maps a given task parallel to taskA to and ordered set
    // of tasks that are parallel to taskA but to which the given task is subsequent.
    Initialize preSequenceMap
    For j = 0 to taskA.parallelSet.count – 1 do
        For k = j + 1 to taskA.parallelSet.count – 1 do
            taskD =taskA.parallelSet.item(j)
            taskE = taskA.parallelSet.item(k)
            If not taskD.parallelSet.contains(taskE) then
Block 3 {       // All tasks that start a fork must be parallel to each other. If there are
                // tasks that are subsequent to each other, these will be appended to their
                // preceding node in the next iteration.
                If not excludedFromFork.contains(taskE) then
                    excludedFromFork.add(taskE)
            If taskD.node is null then
                If not preSequenceMap.containsKey(taskE.identifier) then
                    // Initialize an ordered set for storing all tasks from the parallel
                    // set of taskA, to which taskE is in sequence.
                    Initialize preSequenceTasksSet
                    preSequenceTasksSet.add(taskD)
                    // Add the set to the map for taskE – map.add(value, key).
                    preSequenceMap.add(preSequenceTasksSet, taskE.identifier)
                    nestedForksSet.add(preSequenceTasksSet)
                Else
                    // A set with pre-sequence tasks for taskE already exists – get the
                    // set through map.item(key).
                    preSequenceTasksSet = preSequenceMap.item(taskE.identifier)
```

```
                        If not preSequenceTasksSet.contains(taskD) then
                            preSequenceTasksSet.add(taskD)
                For each taskF in taskD.parallelSet do
                    For each taskG in taskA.parallelSet do
                        If not taskF.equals(taskG)
                        and also not taskF.parallelSet.contains(taskG)
                        and also taskG.node is null then
                            If not preSequenceMap.containsKey(taskF.identifier) then
                                Initialize preSequenceTasksSet
                                preSequenceTasksSet.add(taskG)
                                preSequenceMap.add(preSequenceTasksSet, taskF.identifier)
                                nestedForksSet.add(preSequenceTasksSet)
                            Else
                                preSequenceTasksSet = preSequenceMap.item(taskF.identifier)
                                If not preSequenceTasksSet.contains(taskG) then
                                    preSequenceTasksSet.add(taskG)
                        If not taskF.equals(taskA)
                        and also not taskF.parallelSet.contains(taskA)
                        and also taskA.node is null then
                            If not preSequenceMap.containsKey(taskF.identifier) then
                                Initialize preSequenceTasksSet
                                preSequenceTasksSet.add(taskA)
                                preSequenceMap.add(preSequenceTasksSet, taskF.identifier)
                                nestedForksSet.add(preSequenceTasksSet)
                            Else
                                preSequenceTasksSet = preSequenceMap.item(taskF.identifier)
                                If not preSequenceTasksSet.contains(taskA) then
                                    preSequenceTasksSet.add(taskA)
            For each taskD in taskA.parallelTasks do
                If not excludedFromFork.contains(taskD) then
                    // Add to the fork only the tasks that are not marked for exclusion and not
Block 4             // already processed.
                    If taskD.node is null then
                        parallelSetForFork.add(taskD)
                Else
                    // Exclude sequential tasks also from the nested forks collections.
                    For each preSequenceTasksSet in nestedForksSet do
                        If preSequenceTasksSet.contains(taskD) then
                            preSequenceTasksSet.remove(taskD)
        If parallelSetForFork.count > 0
            // taskA should be added to a fork along with its selected parallel tasks.
            parallelSetForFork.insert (0, taskA)
            // Adjust the sets in the nested forks set. Remove pre-sequence forks sets with less
            // than two elements as forks are addend only for two or more parallel tasks. Remove
            // duplicate pre-sequence fork sets – such may occur if there are more than one tasks
            // in a sequence to a nested fork, i.e. more than one tasks that are parallel to taskA but
            // subsequent to two or more other tasks that are parallel to taskA. Two sets are
            // considered as equal if they have the same number of elements and contain the same
            // elements independently of the elements' order.
            Integer i = 0
            While nestedForksSet.count > 0 and also i < nestedForksSet.count do
```

```
            // Nested forks are created only for two or more tasks.
            If nestedForksSet.item(i).count < 2
            or else nestedForksSet.count > parallelSetForFork.count
            or else nestedForksSet.equals(parallelSetForFork) then
                nestedForksSet.removeAt(i)
                Continue while
            Integer j = i + 1
            While nestedForksSet.count > 0 and also j < nestedForksSet.count do
                // Compare nested fork sets and remove duplicates.
                If nestedForksSet.item(i).equals(nestedForksSet.item(j)) then
                    nestedForksSet.removeAt(j)
                    Continue while
                j += 1
            i += 1
        // If more than one pre-sequence task sets are contained in the nestedForksSet, then
        // multiple nested forks will be added. The top-most fork contains the largest number
        // of nodes to which a further node comes in sequence. So, sort the nestedForksSet in
        // descending order starting from the pre-sequence task set with the most elements.
        Sort nestedForksSet
        // Call Algorithm 6.9 for parallel flow generation based on the detected parallel nodes
        // and pre-sequence sets.
        Initialize forkNode = generateParallelFlow(parallelSetForFork, nestedForksSet,
        previousNodeInBlock)
        lastCreatedNode = forkNode.joinNode
    Else
        // Create a single node for taskA. The node type depends on the structure of taskA in
        // the task delegation graph and may be an atomic task node or an initial node.
        Create nodeA
        // Set references between ad-hoc task instance and workflow task node.
        nodeA.task = taskA
        taskA.node = nodeA
        lastCreatedNode = nodeA
        If not previousNodeInBlock is null then
            nodeA.parentFork = previousNodeInBlock.parentFork
    If not previousNodeInBlock is null then
        // Adjust edges - the edge e_p is the edge with init(e_p) = previousNodeInBlock where
        // init and ter are respectively the initial and terminal nodes for a directed edge.
        Create edge e_b = {lastCreatedNode, ter(e_p)}
        ter(e_p) = lastCreatedNode
```

Algorithm 6.9: Parallel flow generation.

Require:

1) *parallelSetForFork* is an ordered set containing the parallel tasks for the generated fork.
2) *nestedForksSet* is an ordered set containing ordered sets of tasks that are parallel among each other but to which a further task from a larger parallel set is subsequent. The contained ordered sets are used to generate nested forks.
3) *previousNodeInBlock* is a node, to which the generated node sequence is concatenated.

```
Create a forkNode
// Associate the created fork to its parent fork – for nested forks this parent fork is
```

```
// previousNodeInBlock which is passed in the input as discussed later on.
If type of previousNodeInBlock is fork then
    forkNode.parentFork = previousNodeInBlock
Else
    forkNode.parentFork = previousNodeInBlock.parentFork
Compute branch offsets (layout)
// Initialize a set for temporarily storing all nodes, that need to be bound to the join node for
// the created fork.
Initialize parallelNodesInForkSet
// Generate the parallel flow.
For each taskP from parallelSetForFork do
    // Skip tasks for which workflow task nodes have already been created.
    If not taskP.node is null then
        Continue for
    Integer i = 0
    While i < nestedForksSet.count do
        // Get an ordered pre-sequence tasks set within the overall parallelSetForFork. The
        // pre-sequence tasks may need to be added to a nested fork, to which a further node
        // in the parallelSetForFork is in sequence.
        Initialize parallelSetForNestedFork = nestedForksSet(i)
        // Check if the currently processed taskP is contained in a pre-sequence set.
        If parallelSetForNestedFork.contains(taskP) then
            // The tasks for a nested fork are processed in an iterative call.
            nestedForksSet.removeAt(i)
            // Call Algorithm 6.9 for parallel flow generation based on the pre-sequence
            // tasks for the nested fork.
            Initialize nestedForkNode = generateParallelFlow(parallelSetForNestedFork,
            nestedForksSet, forkNode)
            // Bind the nested fork in the graph.
            Create edge e_f = {forkNode, nestedForkNode}
            parallelNodesInForkSet.add(forkNode.joinNode)
            // If a nested fork is generated, the node for taskP has been created in an iterative
            // call to the algorithm for parallel flow generation and the processing of the
            // overall parallelSetForFork can continue.
            Continue for
        i += 1
    // Create a task node for taskP. The node type depends on the structure of taskP in the
    // task delegation graph and may be an atomic task node or an initial node.
    Create nodeP
    // Set references between tasks and workflow nodes (cf. Figure 4.2).
    nodeP.task = taskP
    taskP.node = nodeP
    nodeP.parentFork = forkNode
    Create edge e_f = {forkNode, nodeP}
    parallelNodesInForkSet.add(nodeP)
Create joinNode
forkNode.joinNode = joinNode
// Associate the created join to its parent fork.
If type of previousNodeInBlock is fork then
    joinNode.parentFork = previousNodeInBlock
Else
```

joinNode.parentFork = previousNodeInBlock.parentFork
// Create the edges to form the parallel paths from the fork to the join node.
For each nodeP in parallelNodesInForkSet **do**
Create edge e_j = {nodeP, joinNode}
Return forkNode

The functioning of Algorithm 6.8 for workflow block generation is exemplified in the following by referring to the task ranges and sequence flows given in Figure 6.9. In the following *O* denotes the used ordered evaluation set.

Example 1: Figure 6.9 (g): $O = \{A_j, A_x, A_k, A_p\}$. This example applies also for the tasks with ranges as shown in Figure 6.9 (a) after a *split tail* consolidation. First A_j is handled with a parallel set $P_j = \{A_x\}$ and produces a fork with two parallel tasks A_j and A_x. After this processing step workflow nodes a_j and a_x are available. Thus A_x is skipped in the next iteration step. Then A_k is processed. As a node a_x already exists for A_x, Block 1 from Algorithm 6.8 sets a_j as *previousNodeInBlock* for A_k. Further the parallel set for A_k is $P_k = \{A_x, A_p\}$. As A_p is not in the parallel set of A_x, A_p is marked for exclusion from the fork in Block 3 of Algorithm 6.8. As A_x has an assigned node a_x it is not added to the parallel set for the fork in Block 4 of Algorithm 6.8. As a result A_k is exported as a single task (no fork), subsequent to A_j. Then A_p is processed. As A_p does not have parallel tasks, Block 2 of Algorithm 6.8 takes effect where *lastCreatedNode* is a_k (set in the previous iteration). The parent fork for a_k is found and the corresponding join node is set as a *previousNodeInBlock*. A_p is exported as a single node, subsequent to the found join node, resulting in a sequence flow as shown in Figure 6.9 (g).

Example 2: Figure 6.9 (c): $O = \{A_j, A_x, A_k, A_p\}$. This example applies also for the tasks with ranges as shown in Figure 6.9 (a) after a *merge tail* consolidation. First A_j is handled with a parallel set $P_j = \{A_x\}$ and produces a fork with two parallel tasks A_j and A_x. After this processing step workflow nodes a_j and a_x are available. Thus A_x is skipped in the next iteration step. Then A_k is processed. As a node a_x already exists for A_x, Block 1 from Algorithm 6.8 sets a_j as *previousNodeInBlock* for A_k. Further the parallel set for A_k is $P_k = \{A_x, A_p\}$. As A_p is in the parallel set of A_x (and vice versa) none of the nodes from the parallel set of A_k is marked for exclusion from the fork in Block 3 of Algorithm 6.8. As A_x has an assigned node a_x it is not added to the parallel set for the fork in Block 4 of Algorithm 6.8. As a result A_k is exported in a fork with A_p subsequent to A_j resulting in a sequence as shown in Figure 6.9 (c). The processing of A_p is skipped as a workflow task node for this task has been created during the processing of A_k.

The introduced algorithms support efficiently complex overlapping task ranges and extensive consolidation. Further examples of their functioning are not provided here as these would expand immensely the volume of the dissertation.

6.2.10 Weights and Accuracy of Derived Workflows

Seeding, evolutionary growth and reseeding (SER) [FGY+04] can improve the accuracy of generated workflow models. If a given task pattern is reused multiple times and the ranges of various resulting task instances overlap in multiple executions, the respective tasks can be considered parallel with a greater certainty. With this respect *weights* for the generated workflow graphs can be considered. For example, let A_j and A_k be the tasks from a task pattern *A*. Let A_j' and A_k' be tasks from a task delegation graph resulting from the application of *A*, and let a_j' and a_k' be the workflow nodes produced respectively for A_j' and A_k' after a process model transformation such that in the derived workflow model $\exists\, e' = \{a_j', a_k'\}$. Let A_j'' and A_k'' be tasks from a further task delegation graph resulting from the application of *A*, and let a_j'' and a_k'' be the respective workflow nodes produced after a process model transformation such that in the derived workflow model $\exists\, e'' = \{a_j'', a_k''\}$. Weights can be set to the resulting edges (*e*' and *e*'') to suggest the accuracy of the computed sequence flow. These weights apply for the transformation of all ad-hoc task instances which are related through evolutionary relationships,

i.e. for A_j' and A_j''. A higher weight for a given edge suggests that the derived control flow relationship between the initial and terminal task nodes of that edge is supported in multiple processes following the same task pattern.

Calculation of weights requires that when a sequence flow is generated for a given task from a task delegation graph, the transformation is performed also for all ancestor/descendant task instances. If the transformed ad-hoc task instance has sub-tasks or delegations, the chosen interpretation of hierarchical decomposition and delegations for the currently processed ad-hoc task are used also for the transformation of the ancestor/descendant task instances. Thereby weights are specific for the currently processed ad-hoc task instance, i.e. if ancestors/descendants have sub-tasks or delegations that are unavailable in the transformed task instance, these sub-tasks and delegations are not considered for computing weights in the current transformation.

6.2.11 Alternative Flows

SER capabilities considered in the task management model (Chapter 4) and in the method for composition of weakly-structured process models (Chapter 5) can enable derivation of alternative flows based on substitution and cancellation of subsequent tasks in different task pattern application cases. The generic concept for derivation of alternative (decisions) is to assemble and sort the evaluations sets of all ancestor/descendant task instances for a given task instance and then to check if there are missing or cancelled tasks or task sequences (ordered sub-sets in the evaluation sets) in one evaluation set that are not missing or cancelled in the other evaluation sets. Such tasks or task sequences could be then added in alternative branches to the normal flow through the respective decision nodes (*XOrSplit*, *XOrJoin*). Thereby the correctness criteria for workflow graphs need to be kept, i.e. there must be no overlapping control flow blocks [RD98, RRD03]. Keeping those criteria is a complex challenge as ancestor/descendant executions may have different task change history, different task ranges and thus differently ordered evaluation sets for task instances originating from the same task pattern. As a consequence, workflow blocks for different ancestors/descendants may not be compatible according to the correctness criteria. The SER mechanisms discussed in the thesis deliver the prerequisites for the derivation of alternative flows. The latter aspect is beyond the scope of the thesis.

6.2.12 Deviation Flows

According to the third challenge for business process management systems (cf. Section 1.2.3) end-users need to be enabled to continually extend and adapt process models to evolving requirements, in the context of use [WJ04]. To respond to this challenge, the thesis suggests extending structured workflow models based on deviations through ad-hoc tasks at runtime.

For the following discussion it is important to stress that the transformation of a task delegation graph to a workflow models generates workflow task models (task nodes). When a new workflow instance is started, these models produce workflow task instances. In the following the term *originating task* refers to an ad-hoc task in a task delegation graph, from which a workflow task model (task node) has been derived through transformation (cf. Figure 4.2). A *deviated workflow task instance* is a workflow task instance, for which deviations through ad-hoc tasks have been defined.

6.2.12.1 Relationships Between Ad-Hoc and Workflow Tasks

When a workflow task node is created from a user-defined, ad-hoc task instance, a relationship is established between the ad-hoc task instance and the derived workflow node (cf. Algorithm 6.8 and Algorithm 6.9). The thesis suggests the following further steps for enabling extension of workflow models with user-defined, ad-hoc tasks for deviations:

- When the workflow is deployed, transfer the *originating task* reference of each workflow node to the workflow engine.

- Enable creation of ad-hoc deviation tasks from a workflow task instance in the workflow client application.
- When an ad-hoc task instance is created for deviation from a workflow task instance, store a reference to the deviated workflow task instance in that ad-hoc task instance.
- Track change history of workflow task instances (states not started, suspended, running, ended etc.) to be able to compute temporal relationships between workflow task instances and ad-hoc task deviations.

Following the above steps, the relationships between ad-hoc tasks and workflow tasks given in the task management model (cf. Figure 4.2) are established, i.e. a workflow task model has an association to an ad-hoc task origin, a workflow task instance has an association to a workflow task model (task node in a workflow model), and a workflow task instance can have multiple associations to ad-hoc task deviations.

6.2.12.2 Evaluation Sets for Workflow Redesign

The extension of structured workflows based on deviation flow is performed, when the original task delegation graph, from which a workflow model has been derived, is transformed again. The relationships discussed above are evaluated to assemble an evaluation set that contains the deviated tasks along with the original ad-hoc tasks used for workflow definition. The evaluation set is assembled as follows:

- Let T be an evaluation set that is assembled during the transformation of a task delegation graph as discussed in Algorithm 6.3 and Algorithm 6.4.
- Let $A_j \in T$ be a *sub-task* or a collaborative task (i.e. *requester* or *recipient task*) from the transformed task delegation graph ($1 \leq j \leq n$ and $n = |\ T\ |$).
- Let $a_{j.k}$ be a workflow task node which has been created from A_j through a process model transformation with $1 \leq k \leq m$ where m is the number of all performed transformations of the task delegation graph of A_j (see Figure 6.10). A task delegation graph can be transformed multiple times, producing multiple workflow models, i.e. an ad-hoc task can have multiple corresponding workflow task nodes.
- Let $i_{j.k.p}$ be an instance of a workflow task node $a_{j.k}$ with $1 \leq p \leq q$ where q is the number of all executions of the k^{th} derived workflow model.
- Let $A_{j.k.p.r}$ be an ad-hoc task created for deviation from $i_{j.k.p}$ with $1 \leq r \leq s$ where s is the number of all deviations from $i_{j.k.p}$.

When the transformation of a task delegation graph to a structured workflow model is performed by including deviation tasks, T is extended with all task instances $A_{j.k.p.r}$ of an ad-hoc task $A_j \in T$ from the transformed task delegation graph. The additional operations for extending the evaluation set and for performing the transformation are explained in the following.

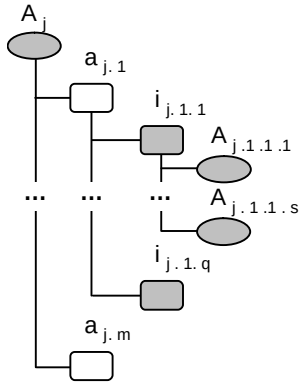


Figure 6.10: Extension of ad-hoc task with workflow task nodes and instances with deviations

6.2.12.3 Task Processing Changes and Task Ranges of Workflow Task Instances

For computing the sequence flow, the ranges of deviation tasks are calculated in the same manner as for all tasks in an assembled evaluation set. However, deviation tasks result from workflow task instances in a structured workflow instance and not from the task delegation graph of the originating ad-hoc tasks from which the model of the deviated workflow has been defined. Thus ranges of deviation tasks cannot be compared with ranges of originating tasks. The comparison is therefore based on the change history and corresponding ranges of workflow task instances.

The thesis considers as **workflow task instance processing changes** such changes, which alter the state of workflow task instances. Basic workflow task instance states such as *not started*, *running*, or *ended* are considered in related workflow literature and supported in different workflow engines [RD98, Dus04]. Updates of task parameters, documents, and user information can also be considered as workflow task instance processing changes and captured depending on the provided functionality of the concrete workflow engine.

The **range of a workflow task instance** is the time between the first and the last workflow task instance processing change.

6.2.12.4 Workflow Redesign Including Deviation Flow

The sequence flow generation for workflow redesign including deviation flow is performed through the following basic operations:

1) *Compute ranges (for originating tasks)*
2) *Sort the evaluation set (for originating tasks)*
3) *Assemble parallel sets (for originating tasks)*
4) *Compute ranges for workflow task instances*
5) *Compute ranges for deviation tasks*
6) *Assemble parallel sets between deviation tasks and originating tasks*
7) *Extend parallel sets between deviation tasks*
8) *Insert deviation tasks in ordered evaluation set of originating tasks*
9) *Consolidate parallel sets for sequence flow generation*
10) *Generate sequence flow*

Operations 1) to 3) correspond to the common transformation procedure where no deviations are considered (cf. Section 6.2.9). If there are deviations, the user is allowed to choose if these should be included in the generated model. Calculations for merging deviation flow may be very heavy-weight if multiple process instances have been executed for a derived workflow model. If the user chooses to include deviation flow during the transformation, operations 4) to 8) are additionally performed.

Operations 4) and 5) compute separately the ranges of workflow task instances and of deviation tasks. Ad-hoc deviation tasks and workflow task instances from the same workflow instance can be consistently checked for sequential order and parallelism based on their ranges.

Operation 6) establishes parallel relationships between the deviation tasks and the originating tasks. As both are created in different processes, relationships between them cannot be established directly. Instead, these relationships are established over the workflow task instances and the respective workflow task nodes produced from an originating ad-hoc task instance as follows.

Let T be an evaluation set, $A_j \in T$ be a task from a transformed task delegation graph, and $A_{j.k.p.r}$ be an ad-hoc deviation task as discussed in Section 6.2.12.2 (cf. Figure 6.10). Let $A_o \in T$ be a further task from the same task delegation graph as A_j ($1 \leq o \leq n$, $n = |\ T\ |$, and $o \neq j$). Let $a_{o.k}$ be an associated workflow task node for A_o with $1 \leq k \leq m$ where m is the number of all performed transformations of the task delegation graph. Let $i_{o.k.p}$ be an instance of the workflow task node $a_{o.k}$, with $1 \leq p \leq q$ and q is the number of all executions of the k^{th} derived workflow model.

The range of an ad-hoc deviation task $A_{j.k.p.r}$ is compared with the ranges of all workflow task instances $i_{o.k.p}$ in the same workflow instance (note that the index k of the derived workflow

model, and the index p of the workflow instance are the same for the deviated task $A_{j.k.p.r}$ and for the workflow task instance $i_{o.k.p}$). If the ranges overlap, the originating ad-hoc task A_o is added to the parallel set of $A_{j.k.p.r}$ and vice versa.

Operation 7) establishes parallel relationships between deviation tasks from a single workflow instance. The ranges of the deviation tasks of each workflow task instance are compared among each other and with the ranges of all other deviation tasks in the workflow instance, i.e. the range of each task $A_{j.k.p.r}$ is compared with the range of each deviation task $A_{j.k.p.x}$ ($1 \leq x \leq s$ and $x \neq r$) and with the range of each deviation task $A_{o.k.p.y}$ ($1 \leq y \leq u$) where s and u are respectively the number of deviations from workflow task instances $i_{j.k.p}$ and $i_{o.k.p}$. Tasks with overlapping ranges are added to each others' parallel sets.

Operation 8) iterates over all deviation tasks and checks their range start time. According to this time, it is determined, after which workflow task instance the deviation task was actually executed. If the range start time of a deviation task $A_{j.k.p.r}$ is subsequent to that of a workflow task instance $i_{j.k.p}$, then $A_{j.k.p.r}$ is considered subsequent also to the corresponding originating task A_j. Thus this operation results in a single ordered evaluation set, including the originating tasks and the deviation tasks.

Operation 9) consolidates the assembled evaluation set. Additional consolidation is performed, when ad-hoc deviation tasks from multiple workflow instances are included in the evaluation set. The workflow instances may originate from the same or from different workflow models that are derived from a given task delegation graph. Temporal relationships between ad-hoc task instances and workflow task instances can be evaluated only in a single process instance (ad-hoc process or a workflow instance). Thus Theorem 5 and Theorem 6 and the respective workflow graph correctness criteria in Theorem 3 and Theorem 4 are kept only for relationships between tasks in the same process instance. When ad-hoc tasks from different process instances are merged in a single evaluation set, the consolidation needs to consider also relationships according to Theorem 3 and Theorem 4. Hence, consolidation is additionally performed for each set of tasks A_j, A_y, A_k in the evaluation set, for which the following is true (cf. Theorem 3 and Theorem 4):

- A_k is *parallel* to A_j and the index of A_k in the evaluation set is greater than that of A_j, i.e. if the parallelism between both tasks is removed, A_k will be subsequent to A_j.
- A_y is *subsequent* to A_j
- A_k is *subsequent* to A_y

The following consolidation options are provided:

- **Merge head** consolidation adds parallelism between A_j and A_y.
- **Merge tail** consolidation adds parallelism between A_y and A_k.
- **Split** consolidation removes parallelism between A_j and A_k.

The *merge* consolidations given above extend the parallel relationships and consider a merge direction analogously to the conventional consolidations on task ranges (cf. Section 6.2.9.1.1). The *split* consolidation checks recursively for other inconsistencies and can result in multiple splits (cf. Section 6.2.9.1.2). The consolidation procedure is given at the end of Algorithm 6.5.

Operation 10) performs workflow block generation as discussed in Section 6.2.9 by using the assembled and consolidated evaluation set and parallel sets.

6.3 Document Flow Transformation

Different process modeling notations support different association of data flow in process models. BPMN [OMG06] for example enables association of artifacts. Other process modeling approaches enable overlay of data flow schemas with control flow [RD98]. The handling of data flow in workflow models and instances is implementation specific and depends on the process

modeling language which is chosen for the workflow derivation and on the used workflow engine. The thesis focuses on the generic conceptual aspects that are relevant for the transformation of task delegation graphs to structured workflow models.

Task delegation graphs emerge from ad-hoc task management and collaboration in the common users' working applications. An intrinsic aspect thereby is not to require end users to learn new application environments and to change their working practice. Unobtrusiveness is needed in order to ensure a gentle slope of complexity [MCLM90] for user-driven process composition. Therefore, for capturing the document flow the thesis considers artifacts from user-defined ad-hoc processes, which emerge as common attachments to task instances in the to-do list and to emails for exchange of tasks or deliverables. Two major types of artifacts are considered for derived workflow models – *static (template) artifacts* and *dynamic artifacts*. The thesis suggests enabling the user to set the type of an artifact during the transformation of an ad-hoc task instance or through explicitly editing derived workflow task models after the transformation.

6.3.1 Static Artifacts

Static (or template) artifacts are such artifacts, which are used but not changed or produced in business tasks. For example, a static artifact can be a customer contract template, which is used always in the same manner when a new customer contract needs to be prepared. Thus, this artifact serves as input for a given task, and can produce a different artifact as output. Static artifacts can be also executable scripts. All artifacts in a derived workflow model, for which the type is not set during process model transformation, are marked initially as static artifacts. This artifact type is used as a default type because it preserves all information that is available also in user-defined, ad-hoc task instances.

A static artifact can be an *externally-managed artifact*, an *externalized artifact* or a *locally-managed artifact* (cf. Section 4.6). Thereby, the first two types can be automatically made available also for workflow task instances by providing access of the workflow engine to the global repository infrastructure and retrieving the artifacts from the artifact repositories. This aspect is discussed in details in the Chapter 7.

Association of *externally-managed artifacts* basically requires document management support and depends on the capabilities of the process modeling environment. The latter may support notifications when an externally-managed artifact is updated (on the artifact repository) so that the process modeler can decide whether to preserve the reference to the previous artifact version or to update to the new one. Such notifications can help to manage up-to-date templates in derived workflow models.

Locally-managed artifacts are available in the transformed process, if a transformation is triggered from the users' to do list and such artifacts are available in local task instances. To make these artifacts accessible for workflow instances on enterprise level, during process model deployment the user needs to export them to the global artifact repository by converting them to externally-managed or externalized artifacts.

6.3.2 Dynamic Artifacts

Dynamic artifacts are such artifacts that are generated or changed during a business process. Dynamic artifacts are considered as parameters for workflow task models and are process instance-specific. For example, if a customer contract is needed in a workflow for handling a customer order, the contract is different for each customer.

Dynamic artifacts can serve as input or output for workflow tasks. When a user sets the artifact type to dynamic, they denote that the actual artifact (content) needs not to be associated to the workflow instance during a concrete process execution. Hence, the actual artifact from an originating ad-hoc task instance is decoupled from the workflow task model and replaced with an appropriate parameter. During workflow execution, the user is able to refer to the parameter for

the dynamic artifact and to upload the actual artifact in the respective workflow task instance.

A user can specify dynamic artifact parameters through manual workflow task model editing. If a dynamic artifact needs to be created based on some template, the user can preserve a static artifact from an ad-hoc task instance and additionally create a dynamic artifact for the case-specific artifact content. Parameter setting and artifact associations are implementation specific.

6.4 Transformation of Human Actor Information - Task Assignments

Recall that each task instance has an associated owner (cf. Figure 4.2). As workflow task models are derived from task instances, they can always obtain the information, who has handled a given task. Based on this information, the task assignments can be transferred from the task delegation graph to the workflow model.

Different process modeling approaches provide different possibilities for task assignment. A common concept is to specify role-based assignments [vdAvH02]. The latter study introduces roles as *"functionally-based resource classes"* by considering the human actors or system components that perform a given task as resources. Tasks in task delegation graphs are always assigned to a specific human actor (the owner) based on their email address as a unique user identifier. This restriction results from the fact that the process composition environment is integrated in the users' working applications and uses only the user data available in them. Through this, users are not confronted with role concepts exceeding their knowledge of the working applications. Thus, during derivation of workflow models the default behavior is to assign resulting workflow tasks to concrete users.

Yet, it may be necessary to provide some generalization of task assignments based on roles. Such generalization can be performed by matching user data of ad-hoc task instances with user data from a workflow management system. Data matching can be enabled through usage of shared repository infrastructure between ad-hoc task management and workflow management systems. The repository infrastructure is discussed in more detail in Chapter 7. Matching of user information between the ad-hoc process composition environment and the workflow management system can allow to find appropriate (eventually multiple) user roles for a given ad-hoc task owner. Thus a role-based assignment for a derived workflow task can be specified during workflow model derivation or after that, through explicit workflow editing. Visually, workflow task assignments can be represented as lanes [OMG06] which encompass tasks of various users or user roles.

6.5 Scientific Achievements

This chapter has introduced a method for composition of structured process models through transformation of user-defined task delegation graphs. Computer-supported cooperative work studies consider embedding structure in ad-hoc operations [Ber00, Bar01]. On the other hand, workflow management and business process management studies focus on embedding flexibility in structured workflows [vdABV+99, Jor04, vdAWG05, Ber05]. The latter aspect is in the focus of current initiatives for developing new human task standards for formal process definition languages such as the WS-human task standard [AAD+07]. However, transformation of ad-hoc processes or process fragments in the form of task hierarchies or task delegation graphs to structured workflow models is not addressed by any scientific or commercial research.

Transformation of task delegation graphs to structured workflows through the provided method supports end-user driven business process management from process emergence to formal process design [Ver04]. Through the derivation of structured workflow models based on

purposeful user actions on task representations in a task management system, the method extends the value of collaborative programming by example [Cyp93, Lie01] towards the automation of rigidly recurring processes on workflow engines. Further, as formal process models are derived from user-defined, weakly-structured process models, the method fosters process tailoring as collaboration [MM00] between end users, process designers and developers. This helps to reduce inconsistencies between the requirements of end users and the formal model definitions provided by business technology experts [Her00], as the latter can work directly with user-defined process models. Thereby consolidation of a derived process model between business users and business technology experts is performed rather than formal process modeling for scratch.

Concepts for extending structured workflow models based on user-defined, ad-hoc task hierarchies from runtime deviations are further provided in the presented method. These concepts extend the concepts for task delegation graph transformation to structured workflows and support user-driven process redesign [Rei05]. The concept extensions enable iterative refinement of derived structured process models by end users, in the context of use [WJ04]. This extends the SER [FGY+04] capabilities of the presented approach in the domain of operational business processes.

6.6 Summary

This chapter has described a method for transformation of weakly-structured task delegation graphs to structured workflow models. The method enables different interpretation of hierarchical task decomposition and delegation flow during the process model transformation. This facilitates the consolidation of weakly-structured process models during their formalization.

The method establishes temporal relationships between ad-hoc task instances based on their change history. Different consolidation options are provided to resolve inconsistencies of the resulting temporal relationships regarding the correctness criteria of formal workflow graphs. These consolidation options enable flexible interpretations of temporal relationships between ad-hoc tasks towards producing well-formed control flow blocks.

The introduced method further enables continuous extension of workflow models through ad-hoc deviations from workflow task instances during workflow execution. Through this end users are enabled to extend process models in the context of use for evolving business requirements.

The method further describes the transformation of document flow from task delegation graphs to structured workflow models. Document flow is transformed through two major artifact types – static and dynamic. Static artifacts allow references to template data. Dynamic artifacts provide process instance specific data for derived workflow tasks.

Transfer of human actor information in terms of task assignments has been further presented. Assignments to concrete users can be directly transferred to workflow task models based on ad-hoc task owner information. Role-based assignments can be further produced by matching user data from ad-hoc tasks to user data in the concrete workflow management system that is used for operational process support. The latter aspect is enabled through a shared infrastructure between the ad-hoc processes composition environment and the workflow management system as discussed in Chapter 7.

CHAPTER 7: Holistic Concept for End-User Driven Business Process Composition

This chapter presents a holistic concept for end-user driven business process composition, which composes the task management model and the process composition methods into a seamless overarching method and architecture for the composition of weakly-structured and structured process models. A generic architecture for enabling composition of weakly-structured process models is described. This architecture supports the introduced task management model and enables the method for composition of weakly-structured process models. The architecture is extended to enable interrelations between the ad-hoc process composition environment and a workflow management system for process automation. The extended architecture enables the method for transformation of task delegation graphs to structured workflow models and the extension of structured workflow models based on ad-hoc task deviations.

7.1 Composition of Weakly-Structured Process Models

This section presents a seamless overarching method and architecture for composition of weakly-structured process models based on the concepts from Chapter 4 and Chapter 5. To clarify the underlying holistic concept, an architecture for end-user driven business process composition based on personal task management is introduced. The architecture responds to the requirements from the empirical studies and the basic end-user development concepts (cf. Table 3.1). It is discussed in the following to clarify how process composition is enabled on system level (cf. [SSFM08b]). The term *office applications* used in the following is a conceptual term and does not explicitly refer to Microsoft Office or imply features offered by this environment.

The architecture is shown in Figure 7.1. It is a three-tier architecture [Mül05] consisting of client, server and persistence layers. The client layer contains the components for personal task management. The server layer comprises the components that provide the tracking functionality, the overall repository access and data retrieval, and the business logic (e.g. notifications handling). The persistence layer encompasses the runtime data storage for task tracking and the user, artifact, and task pattern repositories. The roles of the different components in the overall methodology for business process composition are explained in more details in the following.

7.1.1 Personal Task Management

Personal task management is enabled through the client-side components of the process composition environment which are integrated in the office applications of the end user. According to the findings from the preliminary empirical studies (cf. Chapter 2), email and to-do lists with tasks are the mostly used tools for the management of day-to-day activities. This observation is largely supported in related literature [BDHS03, BDG+04, BDH+05, SIT06]. Thus email and to-do lists are considered as the primary target integration environments for end-user driven process composition.

7.1.1.1 Office Applications Integration Layer

The task management system of the process composition environment is coupled to the office applications over an Office Applications Integration Layer. The integration layer enables programming by example [Cyp93, Lie01] of weakly-structured process models by capturing and recording the users' task management activities that are performed in the personal workspace, i.e. in the common working applications.

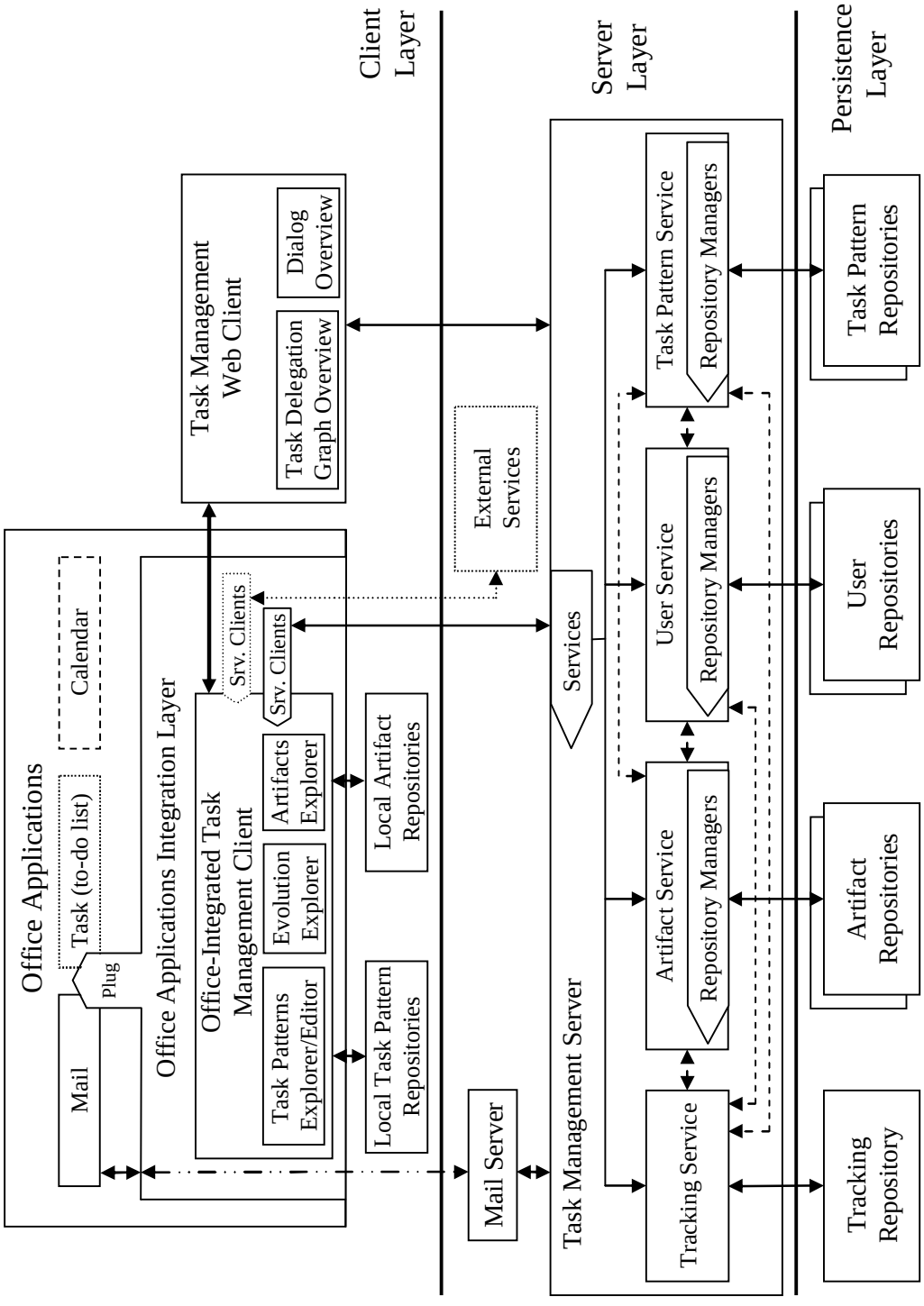


Figure 7.1: Architecture for end-user driven composition of weakly-structured process models

The integration layer enables the usage of email for the purposes of the task management system, by serving as a proxy and enabling email pre-formatting of outgoing emails and appropriate handling of incoming emails with task-related content. Pre-processing and post-processing of emails is common in computational email environments [Bor92, ADMG97] and has been discussed in Chapter 5.

If task management functionality is included to some extent in the office applications environment, the integration layer can use it by enabling tracking of task-related operations (e.g. edit, create, delete). The integration layer provides also functionality to embed the user interface of the task management system in the office applications environment. Tracking and embedded support can be realized by using plug-ins into the office applications. Usage of plug-ins is proposed in related literature on evolving workflows for ensuring unobtrusive process composition support [Her00].

7.1.1.2 Office-Integrated Task Management Client

The Office-Integrated Task Management Client holds the complete presentation logic for the task management system within the selected office integration environment. The main components of the task management client are discussed in the following by referring to the requirements for end-user driven business process composition from Chapter 2.

A **personal to-do list** (R1) is provided in the Office-Integrated Task Management Client, which may be an extended version of the to-do list, provided by the office application environment. In Figure 7.1 this case is considered and therefore no additional to-do list is displayed in the boundaries of the Office-Integrated Task Management Client. Depending on the chosen integration environment, the to-do list functionality may need to be extended to enable hierarchical task decomposition, attachment of artifacts, and indication of task progress information and task states.

A **task pattern explorer/editor** component is further provided for task pattern management. Creation, retrieval, adaptation and search of local and global task patterns and task pattern repositories is enabled in this component. Thus, this component addresses (R4) and supports seeding, evolutionary growth and reseeding (SER) [FGY+04] of weakly-structured process models (cf. Table 3.1).

A **task evolution explorer** component is considered for inspecting the relationships between running ad-hoc processes and corresponding task patterns (R5) and between different task pattern variations (R6). Thus this component provides enhanced analytical capabilities on task instances and task patterns supporting task analysis in the context of SER [FGY+04].

An **artifact explorer** component is considered for supporting artifact management and analysis. It enables users to add externally-managed artifacts, to explore their version and history, and to explore artifact references in tasks through querying data from the server.

The functionality of the components discussed above is supported through task management service clients. These are responsible for tracking task related actions on the task management server and executing updates and queries on the remote repositories. Additional service clients can be plugged in the Office-Integrated Task Management Client to execute task-related operations on external systems e.g. to trigger transactions in a workflow or Enterprise Resource Planning (ERP) system. A simple approach for binding such external services in task instances and task patterns is through attaching executable scripts as artifacts.

The thesis suggests that through separating different functionalities related to composition of weakly-structured process models to different functional components, a gradual user involvement and thus a gentle slope of complexity [MCLM90] for end-user driven process composition can be achieved. For example, a user may wish only to organize their personal activities and may not be interested in managing or analyzing overall processes. In this case the user can manage ad-hoc tasks in their to-do list without engaging with other components. If a user wishes to save a task structure for reuse or to reuse an existing one, they need to engage with the task pattern explorer.

Users who wish to analyze best-practice evolution or to what extent a current ad-hoc process follows an initial guideline can use the task evolution explorer. Users who wish to declare externally-managed artifacts or to analyze processes based on associated documents in tasks, can use the artifact explorer. Thus a user is enabled to extend their expertise with the existing task management applications by using different components depending on their current needs. This approach is followed for all system components in the client layer.

7.1.1.3 Local, Non-Distributed Storage of Process Information

Ad-hoc task instances may hold information that a user considers confidential. This information can reside in task instances or in contained artifacts. To address such confidentiality, on the one hand enhanced authentication and authorization can be considered for tracked task instances and for artifacts and task patterns in the global repositories. On the other hand, local, non-distributed storage of process information can be enabled. To support the latter aspect the users are allowed to switch off the tracking functionality for specific task instances in order to preserve them as confidential tasks within the local workspace. If such tasks have been delegated previously in the context of a collaborative process, disabling of the tracking functionality interrupts the emerging task delegation graphs at the respective ad-hoc task instance nodes.

Local storage of task patterns is further considered to enable reuse of best-practices on personal level. In this case task patterns are stored to local task pattern repositories and reside in the local task pattern scope (cf. Section 5.2.1).

Local storage of artifacts is considered for locally-managed, non-externalized artifacts. Such artifacts are accessible only in task instances in the local to-do list and in local task patterns.

An additional aspect that is considered for storing task patterns and artifacts locally is to provide caching mechanisms for the Office-Integrated Task Management Client. The latter can store local copies of once retrieved globally-accessible task patterns and artifacts in order to avoid communication overhead for repeated retrieval. Caching raises additional issues related to the synchronization of locally cached entities with the global repositories, which are implementation specific and are not discussed in the thesis. Local repositories can be implemented differently, and can be e.g. file-system based or database based.

7.1.2 Process Overview

For providing enhanced transparency into evolving collaborative processes beyond the personal workspace (R3), the thesis suggests using a web-based client. This client is not bound to a concrete (office) application environment and thus is capable of providing a globally accessible process overview. Two components are considered for the overview functionality in this client.

A **task delegation graph overview** represents the personal task hierarchies of all participants in a collaborative process, which are interconnected based on task delegation over email. This overview provides access to all attributes of task instances such as status, percent complete, due date etc., and to globally stored (externally-managed or externalized) artifacts. Thus, this overview enables users to evaluate work distribution and to identify potential bottlenecks and optimization possibilities.

A **dialog overview** aggregates all messages of a task-related dialog. This overview helps users to navigate through messages for task delegation and exchange of deliverables. The overview can thus reduce the user effort for searching task-related emails in email folders, which constitutes a significant amount of the time losses related to personal task management [BDG+04].

The web-based client is accessible from the personal task management environment, i.e. from tasks in the to-do list and task-related emails. In advanced implementations the web-based client may include functionality for dynamic task changes in shared-accessible task hierarchies (cf. [Ber00]). Such changes need to be accordingly reflected in the task instances in the local user workspaces and supplied with notifications to preserve the consistency of the overall process.

To sum up, a web-based overview is provided for task delegation graphs (ad-hoc processes) and task-related email exchange (task dialogs). This overview functionality enables users with higher domain expertise, i.e. local developers or also *"tinkerers"* [MCLM90], to inspect evolving collaborative processes beyond the personal workspace and to identify problem areas and optimization possibilities. Hence, the overview provides enhanced decision support for evolving ad-hoc processes and through this also incentives to the end users to extend their skills with conventional applications for personal task management and collaboration.

7.1.3 Capturing Processes

Collaborative programming by example [Cyp93, Lie01] of overall enterprise processes is realized through tracking user actions on personal task management and aggregating the data into weakly-structured task delegation graphs on a central server instance. The captured tasks, artifacts and human actor information are considered as reflecting different aspects of emerging process models, i.e. control flow, document flow and task assignments. User-defined process data can be then reused to reconstruct captured process examples, i.e. through the extraction and application of reusable task patterns. To enable enhanced analysis of emerging business processes from different perspectives and to facilitate data reuse, the thesis suggests data dissemination through different services and repositories for the different entities – tasks, artifacts and human actors. The services and repositories enable interrelation of the different entities and their aggregation into different views for enhanced process analysis and decision support.

7.1.3.1 Mail Server

The mail server is implicitly included in the architecture shown in Figure 7.1 as the exchange of tasks is realized over email, i.e. this component is used by the office applications email client and could be e.g. a Microsoft Exchange server. Furthermore, email can be used for the propagation of notifications from the back-end to the clients. Thus a bidirectional relation from the task management server to the mail server is depicted in Figure 7.1. However the presented architecture does not rely exclusively on computational email as known email-based workflows [ADMG97] but utilizes remote services to increase performance and extensibility. These services are comprised in the process composition server application and are used for tracking, notifications and data retrieval. Computational email is used by the clients to update the local task information. Local updates that have global effect, e.g. cancellation or completion, are propagated to all recipients over the service infrastructure as discussed in Section 5.1.5.2.

7.1.3.2 Process Composition Middleware

The process composition middleware is provided through a server application that comprises the services for handling task instances, artifacts, human actors, and task patterns. The middleware connects the client applications with the remote repositories.

The **tracking service** updates the persistent state of tasks on the server when these are created or updated on the client. The service handles additionally the collaborative flow. All task-related email exchange is stored in dialog instances, and associated to the appropriate requester and recipient tasks as discussed in Chapter 5. All messages are available in the tracking repository with text and attachments. The attachments are replicated to a remote artifact repository over the artifact service. The tracking service feeds also owner and recipient data in the user repository over the user service. Notifications on task instance changes in the global scope (cancel, complete, delete) are also propagated though the tracking service. Subscription for notifications and events is performed on the tracking service by the respective clients upon startup of the client application. While email-based workflow architectures [ADMG97] use computational mail by enabling self-contained, concurrently executing software processes, the tracking service enables centralized handling of task-related events on the server in order to manage consistently

interrelated tasks of various users.

The **artifact service** provides functionality for the replication of externalized artifacts and for the management of externally-managed artifacts on remote artifact repositories. This includes all update, search, and retrieval functionalities.

The **user service** provides functionality for storing and retrieving user information to and from user repositories.

The **task pattern service** updates global task patterns, enables search in the global task pattern repositories, and delivers global task patterns to the clients. Changes to task patterns which are reused in running processes can be supported with appropriate notifications through the task pattern service. These notifications can allow the user that has reused a task pattern to check how the overall best-practice has been changed over the evolutionary (ancestor) references. The user can estimate whether the changes are applicable also for the running ad-hoc process and adapt the affected task instances accordingly.

All three services: the artifact service, the user service and the task pattern service, handle different repository types (e.g. database or file system based) through appropriate *repository managers*. This is especially important for the artifact and user services, as they can facilitate the interplay between ad-hoc and structured processes as discussed later on in this chapter.

The tracking service and the task pattern service communicate with each other to enable setting and retrieval of ancestor/descendant relationships between tracked tasks and global task patterns. Both services communicate with the artifact service, to maintain associations of artifacts within active (tracked) tasks and global task patterns. All services have access to the user service: (i) the tracking and task pattern services feed task owner and recipient information; (ii) the artifact service feeds author (i.e. external artifact manager) information for externally-managed artifacts.

7.1.4 Data Dissemination and Reuse

For supporting enhanced data reuse between different process instances and models the thesis suggests dissemination of process data to different repositories. These are shown in the persistence layer in Figure 7.1. The persistence layer comprises the task tracking repository and the user, artifact, and task pattern repositories. The depicted repositories can reside on physically different hosts or they can be embedded in a single (database) application. In the first case additional repository mapping functionality is necessary in the respective services for relating the different entities – tasks, users, artifacts and task patterns.

The **tracking repository** stores the data, generated through tracking of client-side user operations in the personal task management environment. The latest state of all user tasks from the personal workspaces is replicated in this repository. The tracking repository stores also all dialogs for task delegation. This repository thus provides the input for the web-based overviews of task delegation graphs and dialogs in the task management web-client.

A **global (remote) artifact repository** holds reusable artifacts. The artifact repository supports artifact versioning for externally-managed artifacts. One or more artifact repositories of the same or different kind, e.g. database or file system based, can be managed in a unified manner through the *artifact service*. It is useful to consider keeping all globally accessible artifacts – externalized artifacts and externally-managed artifacts in the same repository. As both artifact types have similar attributes (cf. Section 4.6), keeping them in the same repository can enable seamless conversion of an externalized artifact to an externally-managed artifact through extending the attributes' set of that artifact.

The **user repository** stores human actor information. User data is added in the following ways: (i) through tracking of evolving tasks in running processes, i.e. task creation feeds owner data, task delegation stores recipient information when the respective message is tracked; (ii) saving of a global task pattern that contains human actor information (owner, suggested recipients); (iii) adding and editing of externally-managed artifact adds author information.

A **global (remote) task pattern repository** holds reusable task patterns. More than one

repository can be maintained within the system. The tracking repository can be considered as an implicit task pattern repository as task patterns can be extracted from captured process instances at any time. However, while task structures in the tracking repository may be changed frequently, i.e. when a user updates tasks in their personal to-do list, the idea of a global task pattern repository is to keep reusable process fragments in a consolidated manner for providing global best-practices.

7.1.5 Facilitating Process Analysis through Multiple Perspectives

[Ber00] describes architecture for ad-hoc process support, where no differentiation between a task runtime repository and task model repository (task pattern) is made, and no explicit artifact and user repositories are suggested. Explicit user and artifact repositories are not considered also in [Jor04] which proposes an architecture supporting explicitly defined, abstract workflow models. The architecture proposed in [Jor04] focuses on the process model enactment and adaptation and does not discuss the aggregation of data for the composition of process models by end users.

The thesis suggests that distributing and interconnecting data in various repositories – tracking, user, artifact, and task pattern repositories through the respective services facilitates data reuse and enables different perspectives on processes. The following perspectives are considered.

A **process instance perspective** for ad-hoc processes is enabled through task instance data from the tracking repository and its aggregation through the tracking service. This perspective describes the end-to-end process flow where tasks contain all relevant artifacts and human actors' information. Tracing of evolutionary relationships between task instances and task patterns through the ancestor/descendant relationships further enables analysis of the handling of ad-hoc tasks in related business cases.

A **process model perspective** for ad-hoc processes is enabled through the task pattern data in the global task pattern repositories. Task patterns can be used for case analysis and consolidation of captured best-practices. Such consolidation is facilitated through the evolutionary relationships provided through ancestor/descendant references.

An **artifact perspective** is enabled through the data in the artifact repositories and through the artifact service. The artifact service and repositories enable enhanced analysis for detection of similar tasks based on the usage of similar artifacts. This analysis can be realized by retrieving and evaluating references to artifacts in task instances and task patterns. The artifact perspective can assist towards document-based proactive information delivery on task instances [HRD+06].

A **human actor perspective** is enabled through the data in the user repository and through the user service. User associations to task instances based on the owner and recipient information enable evaluation of work distribution in running processes. User associations in task patterns enable basic expertise analysis and expertise recommendation. Author (i.e. external artifact manager) associations to externally-managed artifacts enable detection of authorship, expertise, and contributions.

The provided concept for enabling multiple perspectives on task, artifact, and user data has been used to develop analytical tools for supporting knowledge-intensive work [Sch08].

7.2 Process Automation

The introduced architecture for composition of weakly-structured process models is extended to support the method for derivation of structured workflow models from user-defined task delegation graphs (cf. Chapter 6). The extensions are shown in Figure 7.2.

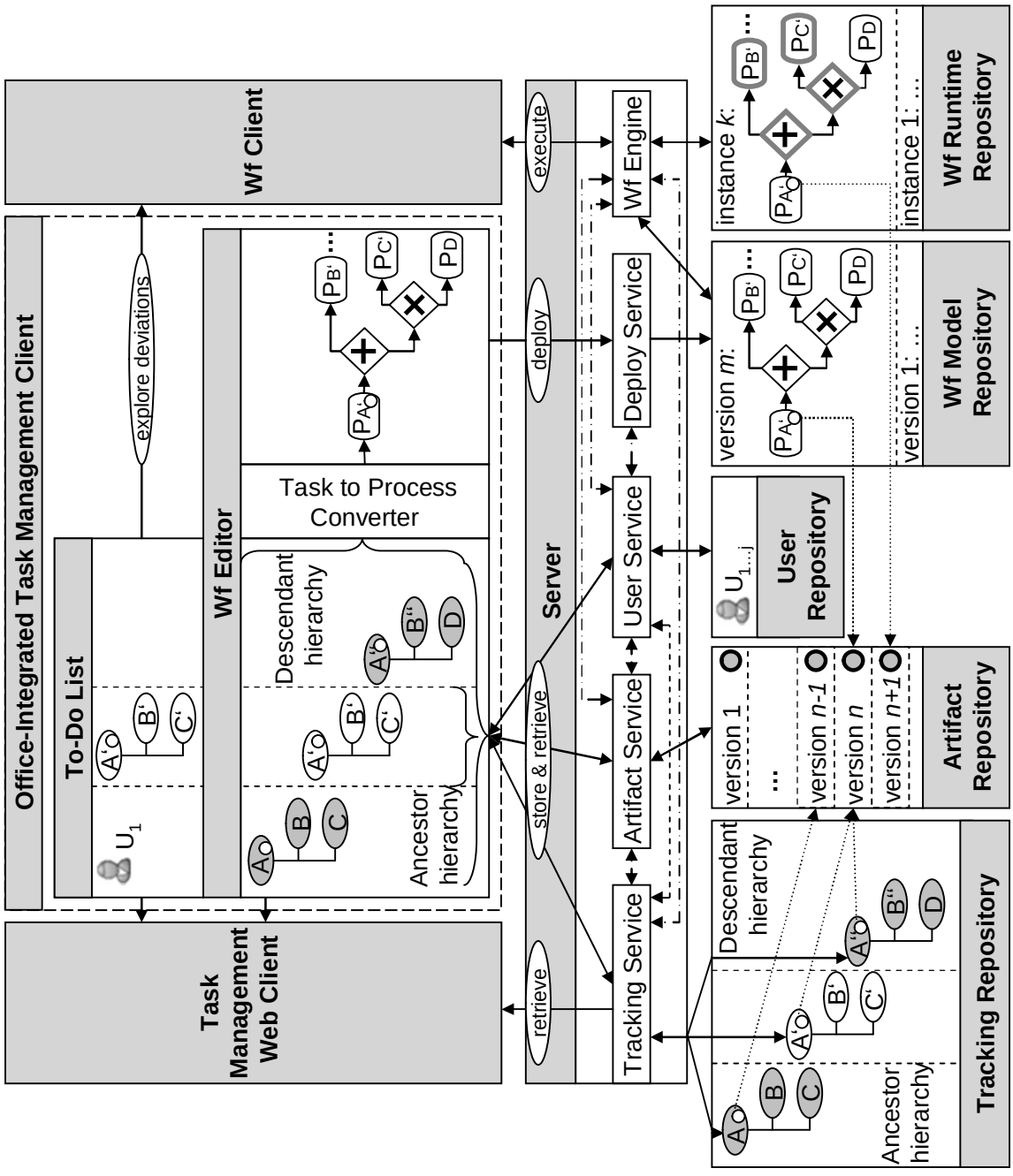


Figure 7.2: Architecture for end-user driven composition of structured process models

A user U_1 manages task instances (*A'*, *B'*, *C'*) in a hierarchical to-do list in an Office-Integrated Task Management Client. The task instances are tracked over the tracking service and replicated in the tracking repository. The tracking repository contains also alternative executions (ancestor/descendant instances) of the given task, resulting from task pattern reuse. Task *A* has been extracted as a task pattern which has been reused, resulting in task *A'*. A further extraction of *A'* as a task pattern and its reuse has resulted in task instance *A''*, where the original task *C''* is replaced with a task *D*.

The white circles with a black outline in tasks *A*, *A'* and *A''* represent local artifact associations to global artifacts residing in the artifact repository. The latter are shown as gray circles with a black outline. No differentiation between externally-managed and externalized artifacts is made here for simplicity.

The dashed arrows in the server layer in Figure 7.2 represent the connections, used during ad-hoc task management, and the chain-dotted arrows – the connections used during workflow management. No concrete workflow engine is assumed in the discussion on the transitions between user-defined and formal process models in the next sections.

7.2.1 Shared Context for Process Tailoring

One of the major research challenges addressed through the thesis is to enable a shared context between user-defined and formal process models. Having such a context is needed to support process tailoring as collaboration [MM00] between end users, process designers and developers.

To enable a shared context, the thesis suggests providing an enhanced editing environment for workflow models within the common users' working applications. In the following this environment is referred to as a *workflow editor*. The workflow editor enables editing of formal workflow models according to the selected modeling notation. This editing includes among other the adjusting of workflow task model parameters and artifact associations, and setting of artifact types. The following aspects are further considered in the workflow editor to enable process tailoring as collaboration.

A **shared representation of user-defined task hierarchies and derived workflows** is provided to the user, where they can establish a direct relationship between an originating ad-hoc task and a derived workflow task model (task node). This relationship enables the end users to see, how their task is represented in a formal workflow model. On the other hand, a process modeler or developer is able to see the original user-defined task instance for a workflow task model, to check its context information, and to see if the developed formal model is correct from business perspective.

To support evaluation of formal models from business perspective, the complete evolution history for a transformed ad-hoc task instance can be retrieved, i.e. all existing ancestor/descendant instances, showing execution of this task in similar cases. These evolutionary relationships can further reveal alternative flows based on substitution, cancellation or deletion of tasks instances in different task pattern application cases. Ancestor/descendant hierarchies can be retrieved in the workflow editor though the tracking service. If a task instance is originating from a task pattern, ancestor/descendants of this pattern can be retrieved through the tracking service over the task pattern service. Both services are interconnected as shown in Figure 7.1. This relationship is not given in Figure 7.2 for simplicity.

A **transformation control** is further provided, where users can adjust the transformation options. The change events for task instances that should be interpreted as task processing changes (cf. Section 6.2.6) are specified through the transformation control functionality. This functionality further enables the user to select whether to export cancelled tasks. The transformation control enables also the selection of export mode options for parent tasks and delegated tasks, thus making use of the different interpretation possibilities for hierarchical task decomposition (cf. Section 6.2.3) and delegation flow (cf. Section 6.2.4). The transformation control further allows selection of consolidation options for resolving inconsistencies between

task ranges and workflow graph correctness criteria (cf. Section 6.2.9). The selections of whether to compute weights for generated sequence flow (cf. Section 6.2.10) and whether to include ad-hoc task hierarchies resulting from deviations in derived workflow instances (cf. Section 6.2.12) are also enabled through the transformation control. Thus, the transformation control allows the user to experiment with the transformation options, to see how the process models are influenced by these options, and to eventually select the most appropriate transformation.

A **transformation rationale** is further considered, which describes the performed transformation in a human-readable form. It can be in the form of a textual description that is supported through some kind of visual representation, e.g. showing task ranges as given in Figure 6.4. This rationale can help the different stakeholders involved in the process transformation to see why a given sequence flow has resulted, e.g. by referring to the change history of user-defined task instances and to the respective ranges.

A **transformation engine** performs the process model transformation according to the specified transformation options, and generates the data for the transformation rationale. In Figure 7.2 this engine is shown as a *Task to Process Converter* component.

Explicit mapping between context information of user-defined task instances and workflow task models is further considered to handle transfer of process-relevant information for which a direct (automatic) transformation is not possible. Such transfer is needed for example, if task assignments for the workflow task nodes need to be specified in a role-based rather than in a user-specific manner. In this case, if the workflow engine and the ad-hoc process composition environment share a common user repository (cf. Figure 7.2), during process model transformation user data of ad-hoc tasks can be extended in workflow task models to match the requirements of the concrete workflow management system. Such extensions can be supported in a semi-automated manner, where the user service mediates between the ad-hoc process composition environment and the workflow management system, and proposes possible matching between available user data of ad-hoc tasks and workflow task assignments. If the owner of an ad-hoc task matches more than one role in the workflow management system, the process modeler is enabled to select to which role to assign the resulting workflow task node, by eventually checking the context information of ad-hoc task instances, the task delegation graph, and relevant dialogs. If different user repositories are used by the ad-hoc process composition environment and by the workflow management system, the user service may not be able to automatically map task assignments. Thus, explicit manual operations will be required in this case to transfer the assignments.

7.2.3 Automation Support with Ad-Hoc Task Interrelation

In **workflow models** generated task nodes ($P_{A'}$, $P_{B'}$ etc., see Figure 7.2) receive references to the respective originating ad-hoc task instances (*A'*, *B'*) and to the used artifacts (in $P_{A'}$). Formal workflow models can be deployed on the server over a *deploy service*, which stores them to a *workflow model repository*. The latter two components are conceptual abstractions that do not refer to a concrete implementation. The deploy service refers to a conceptual service, which is used to deliver a new workflow model to a workflow engine. The workflow model repository denotes the placeholder in a workflow management system, where workflow models are stored.

Workflow instances can be started and managed over a *workflow client*. It communicates with a *workflow engine*, which instantiates workflows from the deployed models and maintains their state in a *workflow runtime repository*. The workflow client, engine, and runtime repository are also conceptual terms that denote generic components for execution of workflow instances. The parallel and exclusive gateways and workflow task nodes $P_{B'}$ and $P_{C'}$ in Figure 7.2 are marked to denote an example flow of a workflow instance.

Artifact references in workflow task instances (in $P_{A'}$) can be used to retrieve available *static artifacts* (cf. Section 6.3.1) from the artifact repository. Different *dynamic artifacts* can be added to a workflow task instance, which stores these in the artifact repository over the workflow

engine. Added dynamic artifacts can be externalized similarly to ad-hoc task attachments. This enables matching of workflow task artifacts with ad-hoc task instance artifacts in the artifact repositories over the artifact service. Such matching can enable detection of similarities between ad-hoc tasks and workflow tasks based on the usage of same or similar artifacts.

Deviations from workflow task instances are supported in the workflow client to enable user-driven extensions of derived workflows (cf. Section 6.2.12). An ad-hoc task instance is created for a workflow task instance upon deviation. This is performed by sending a request for ad-hoc task creation over the workflow engine to the tracking service, which issues an appropriate event to the Office-Integrated Task Management Client.

The event transfers *workflow task* and *process instance identifiers* which are used to map the deviated workflow task instance and the resulting ad-hoc task instance on the server (cf. also Figure 4.2). The mapping is performed through storing the workflow task instance identifier in the tracking repository. This interrelation enables navigation from the workflow task to the task delegation graph of the ad-hoc task and vice-versa in the client applications.

The deviation event further transfers to the created ad-hoc task a *reference to the originating ad-hoc task*, which has been used to generate the deviated workflow task. For example, for $P_{B'}$ a reference to task *B'* is transferred. This allows retrieval of the respective task (*B'*) and all related ancestor/descendant tasks (*B* and *B''*) from the tracking repository. Thus, recommendation for the further handling of the deviation based on task patterns can be provided.

The ad-hoc tasks are decoupled from the workflow and the execution of a deviated workflow task can continue, e.g. if the deviation is an extension to the workflow rather than an exception that requires workflow termination. While the ad-hoc task management server tracks the changes of the deviating ad-hoc task instances, the workflow engine tracks the state of the deviated workflow task instance. This tracking allows evaluation of the temporal relationships between deviating ad-hoc task instances and the respective workflow task instances. After the workflow has ended, the workflow model can be redefined by considering the ad-hoc task hierarchies of deviations in addition to the original tasks, used for workflow definition (cf. Section 6.2.12).

7.3 Scientific Achievements

This chapter has presented a seamless, overarching method and architecture for end-user driven business process composition of both weakly-structured and structured process models.

Gradual involvement of end users in business process composition is provided by the overarching method and architecture for ensuring a gentle slope of complexity [MCLM90] for process tailoring. The gradual involvement is enabled through dividing various functionalities for ad-hoc process support in different functional system components. Thereby end users are enabled to engage with those components that are of interest to them and that can bring immediate benefit for their personal work. Personal task management is enabled through client-side components which are integrated in the to-do list and email applications of end users. The latter applications are considered as common working applications of end users by a large body of research [BDHS03, BDG+04, BDH+05, SIT06] and are thus appropriate for ensuring unobtrusive process composition support. An integration layer is proposed that integrates the process composition environment in the users' working applications. This layer and the comprised components for personal task management enable programming by example [Cyp93, Lie01] of weakly-structured process models by capturing and recording the users' task management activities that are performed in the personal workspace. The integration layer further enables the usage of computational email [Bor92] for pre-processing and post-processing of task-related email messages, which is needed to capture the task-related email exchange and to utilize email as a central tool for end-user driven business process composition.

Related studies on user-centric process support [Ber00, Jor04, HMBR05, HRD+06] do not consider gradual involvement of end users in business process composition through integrated support in the actual end users' working applications. Integrated support for composing evolving workflows is proposed in [Her00]. However, the latter study does not provide concrete concepts or architecture for supporting evolving workflows. Thus, the discussed overarching method and architecture extend current work on user-centric process support with additional concepts for ensuring a gentle slope of complexity [MCLM90] for process composition by end users.

Capturing of data on personal task management of different users and dissemination of this data on a central server infrastructure is enabled through the discussed method and architecture. The data replication on enterprise level enables collaborative programming by example [Cyp93, Lie01] of end-to-end business process models by multiple users, beyond personal workspaces. Different task, artifact, user, and task pattern services are provided by a middleware server application for handling the different task management model entities. Captured data is disseminated respectively to: task, artifact, user, and task pattern repositories. Related architectures for ad-hoc process support [Ber00, Jor04] do not differentiate between a task runtime repository and task model (task pattern) repository, and do not consider explicit artifact or user repositories. The repository structure proposed in the thesis fosters process analysis from different perspectives such as task (process), artifact, user, and task pattern perspectives. The perspectives have been realized in further work [Sch08] to enable enhanced, context-based support for knowledge-intensive work. Through such support incentives can be provided to end users to extend their skills with conventional task management applications towards proactive business process composition based on personal task management.

Derivation of structured workflow models from user-defined data on personal task management towards the automation of rigidly recurring processes is further supported through extensions in the method and architecture for composition of weakly-structured process models. The method for derivation of structured workflow models considers a *shared context* for process tailoring [MM00] between end users, process designers and developers. Furthermore, extensive *user control* over the procedure for transformation of user-defined task delegation graphs to structured workflow models is proposed, as well as assisting functionalities for reasoning about the *rationale* behind the transformations. The extended architecture contains *shared components* pertaining to both, ad-hoc as well as structured process support. These shared components foster data reuse between ad-hoc and structured process instances. The data reuse facilitates the transformation of weakly-structured to structured process models and the redesign of structured process models based on user-defined deviations with ad-hoc task hierarchies. Related methods and architectures for integrating routine and ad-hoc work [Ber00, Jor04] do not consider supporting transformation of user-defined, ad-hoc task hierarchies to structured workflows through shared components between personal task management environments and workflow management systems.

7.4 Summary

This chapter has introduced a holistic concept which encompasses the task management model and the process composition methods into a seamless overarching method and architecture for the composition of weakly-structured and structured process models. The introduced method and architecture propose integrated support in the users' working applications, which allows the users to gradually involve in process composition. Users are enabled to extend their current expertise with working task management and email applications towards process tailoring by using only the components and functionalities that provide immediate benefit to them.

The proposed method and architecture further enable dissemination of process data in different

repositories. The provided dissemination facilitates data reuse and enables process analysis from process-centric, artifact-centric, and user-centric perspectives. These perspectives can be realized by flexibly composing data according to the decision support that is currently needed by an end user. Such analytical capabilities can provide additional incentives to end users to involve in business process composition based on personal task management.

The transition between user-defined task delegation graphs and structured workflow models is supported through shared access of the ad-hoc task management environment and the workflow management environment to components that store task, artifact, and human actor information. This access enables users to work in a shared context between user-defined and formal process models, by referring to common task and process data. The shared infrastructure further enables interrelation between ad-hoc task instances and workflow task models and instances. This interrelation enables incremental extension and refinement of derived workflow models based on ad-hoc task deviations at runtime.

CHAPTER 8: Implementation - Collaborative Task Manager

This chapter describes the implementation of the concepts that have been elaborated throughout the dissertation. The concepts are realized in a tool called Collaborative Task Manager (CTM). This tool enables process-enhanced task management and supports the composition of both: (i) weakly-structured process models for supporting ad-hoc business processes, and (ii) structured workflow models for automation of rigidly recurring processes on a workflow engine. The implementation is used for the evaluation of the concepts as discussed in Chapter 9.

8.1 Basics

The Collaborative Task Manager (CTM) is an email-integrated task management tool, with extensive support for composition, adaptation and reuse of weakly-structured process models, as well as for the derivation and redesign of formal workflow models. All industry partner companies involved in the preliminary empirical studies (cf. Chapter 2) were using Microsoft Outlook as a standard email client. To ensure an integrated support within the common working environment of end users, CTM is delivered as an Outlook add-in, additionally exploiting the fact that tasks and email are provided in the same office application (cf. also Figure 7.1).

The Office Applications Integration Layer (cf. Figure 7.1) comprises classes with proprietary extensions of Outlook email and task items, adding custom properties and a set of event handlers to these items. User actions on CTM tasks and email messages are captured through the event handlers and tracked over web services. All services (tracking, artifact, user, task pattern) are based on the Simple Object Access Protocol (SOAP) [W3C00]. SOAP has been chosen because of its well-established standard, extensive usage, and provided optimization mechanisms. For example the SOAP Message Transmission Optimization Mechanism [W3C05] is used for transferring binary artifacts between the client and the server. The CTM services are comprised in a CTM server application which is based on the Java Enterprise Edition and deployed on a JBoss application server [JBoss].

The local artifact and task pattern repositories are file-system based. All remote repositories are integrated in a single database. Relationships between the different entities of the task management model are realized through foreign-key relationships in the database.

CTM uses a single artifact repository for storing externalized artifacts. This repository is based on a database table with artifact context information such as name, checksum, version etc., and links to actual artifact content (files) on the server file system. Implementation for externally-managed artifacts has not been provided because a document management system was not feasible for the real-life evaluation usage in the partner companies. Limitations regarding the evaluation are discussed in Chapter 9. During the design phase of the CTM system, it has been further considered that explicit document management requires additional knowledge of the respective document management system. Therefore externally-managed artifacts have not been considered as being of particular interest for light-weight process composition from the available, current end users' working applications. Locally-managed, non-externalized artifacts are further not supported, as the focus of the implementation and later evaluation has been set on small user groups where privacy issues were not anticipated by the end users. Thus the provided externalized artifacts are considered as sufficient to validate the artifact associations in the context of end-user driven business process composition.

8.2 Personal Task Management

CTM enables personal task management in a light-weight to-do list (R1) which is integrated in a common users' working environment. The CTM to-do list is shown in Figure 8.1. CTM extends Outlook tasks with functionality for displaying a hierarchical tree structure. The CTM add-in provides additional toolbars for direct access to the main CTM functionalities. The toolbars are context-sensitive and provide only those controls that are applicable to the currently selected task.

CTM enables insertion and removal of tasks and sub-tasks in a task hierarchy in a light-weight manner. Task insertion opens a new Outlook task dialog where the user works with the familiar Outlook task fields. Files can be added to CTM tasks as common Outlook task attachments. To facilitate the transition for email to task, CTM offers the possibility to save an email as a CTM task. Thereby the mail subject, body, and attachments are accordingly applied to the task.

The process and dialog overviews are accessed over a *Process Info* button (upper right corner on Figure 8.1). For delegated tasks, the user can open a *Recipients* drop-down list with recipient information, showing the *recipient info* (cf. Section 4.4.3.3) of all task recipients.

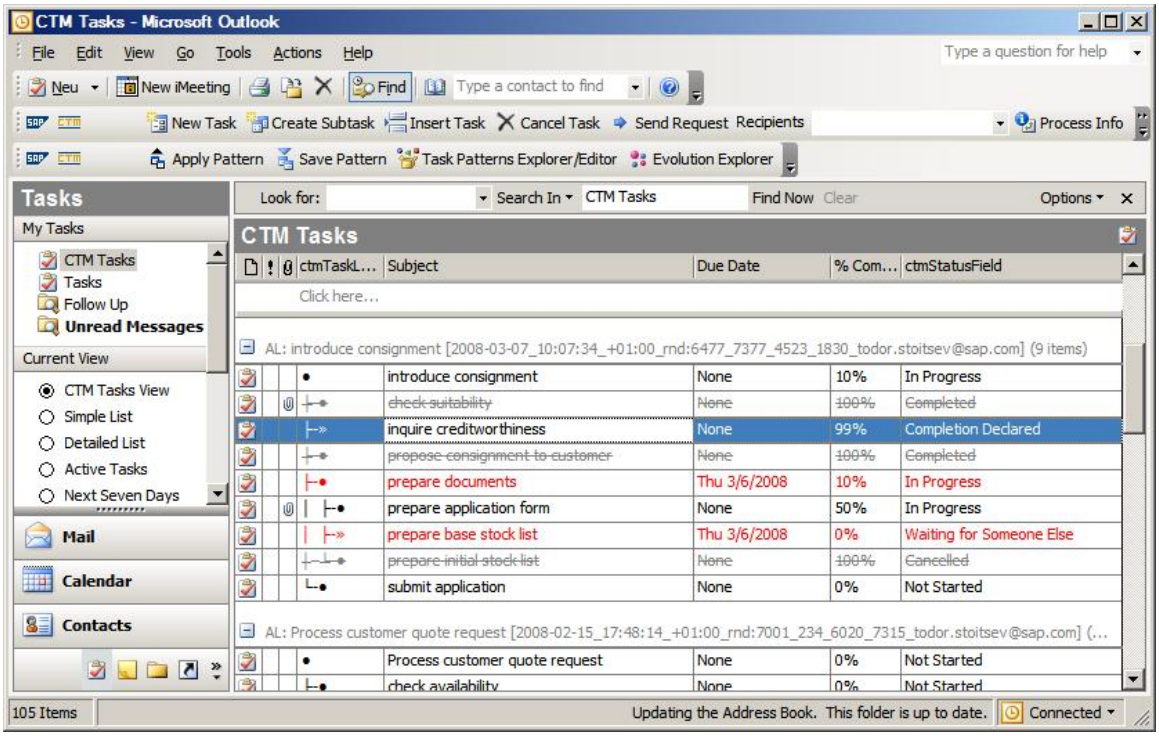


Figure 8.1: CTM to-do list

8.3 Exchange of Tasks and Deliverables

CTM enables email-based person-to-person communication for exchange of tasks and deliverables (R2). A CTM task is delegated through a preformatted *Request* message. When a request is triggered for a given task, all task context information and attachments are transferred to the email. The user can remove the attachments if these are not applicable for the request. Recipients can *Accept*, *Decline* or *Negotiate* the request. The corresponding functionality is provided in a context-sensitive toolbar in Outlook. When the user selects a CTM email, its type is determined based on the embedded meta-information to activate the appropriate controls.

While request/accept/decline are standard actions known also from the exchange of meeting requests in Outlook, iterative negotiations allow additional clarifications on tasks. The actual discourse takes place in the email text, which is independent from the given message type. Thus an open-ended collaboration on tasks is enabled, which does not restrict users to strict speech-act rules. Rigidity of speech acts is a known limitation in speech-acts adoption [But94].

When a request is accepted, and later on completed by a recipient, the latter issues a *Declare Complete* message. Hereupon the requester can respond with *Approve Completion* or *Decline Completion* message. These actions allow negotiation of deliverables, before the final completion of a delegated task.

All email exchange for task delegation and completion declarations is associated to a task dialog and stored on the server. Dialogs can be inspected through a hierarchical process tree-view, where the nodes provide links, opening the exact task and email descriptions, including text and attachments (Figure 8.2).

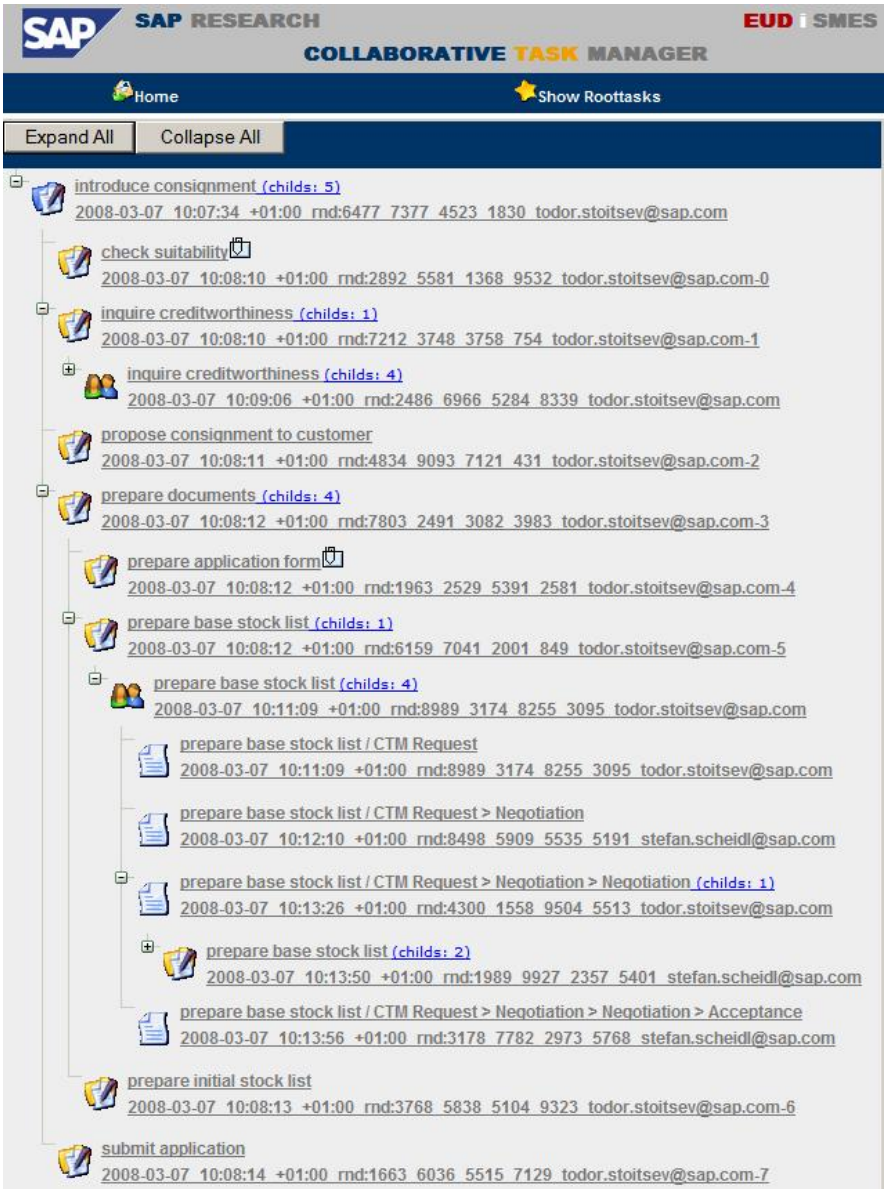


Figure 8.2: Dialog overview

Email overload is a known issue for knowledge workers [BDH+05]. To avoid flooding of the inbox with task-related messages, a *Move CTMs* button is provided in the CTM toolbar, which moves all task related emails to a special CTM email folder.

The collaborative functionality in CTM is further supported through basic notifications. These update the recipient info in requester task instances. Notifications are provided also for cancellation, completion, and deletion of task instances in the global scope (cf. Section 5.1.5).

8.4 Process Overview and Navigation

In CTM, process models emerge as examples for the actual process execution and comprise the individual to-do lists of all process participants, which are integrated based on the tracked task-related email exchange. Thereby overall process models emerge as task delegation graphs, where the personal tasks of different users are shown in different user containers. The CTM task delegation graph overview is shown in Figure 8.3. The overview is composed from the tracked data on personal task management and task delegation from the tracking repository.

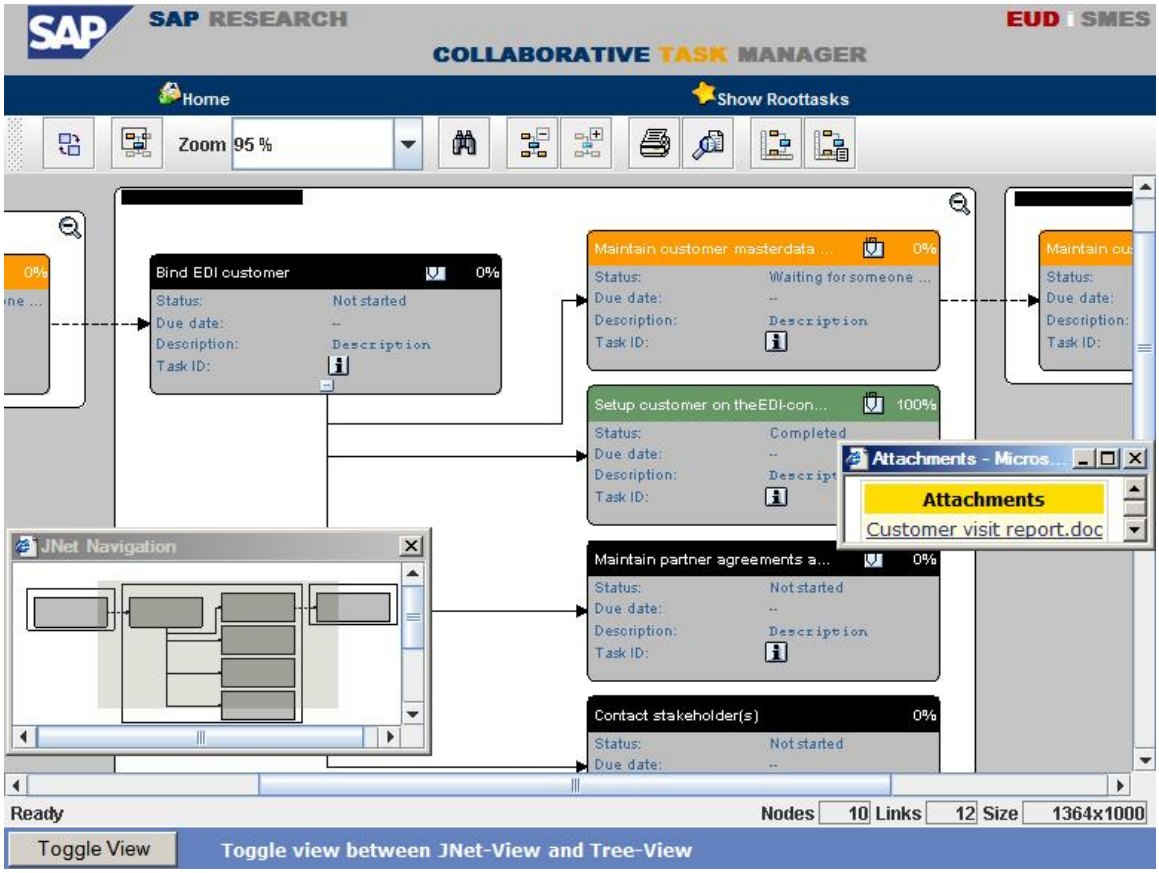


Figure 8.3: Task delegation graph overview

The white, rounded rectangle areas in the background in Figure 8.3 represent user containers. The user email-address is displayed in the upper left corner of each user container (in Figure 8.3 email-addresses are anonymized for privacy reasons). Each user container contains the task hierarchies of a given user that are part of the currently displayed ad-hoc process instance.

The rounded rectangles in the user containers represent task instance nodes. These nodes are direct replicates of task instances that are defined and managed in the personal to-do list of an end user (cf. Figure 8.1). A parent task is connected with its sub-tasks through arrows, which represent the hierarchical decomposition of tasks. Dotted arrows represent delegations. Each change on a task instance in the to-do list is replicated on the central server instance and reflected in the respective task instance node in the task delegation graph overview to keep the process up to date. Status, percent complete and due date attributes are provided in task instance nodes to enable end users to detect approaching deadlines and possible bottlenecks. Task statuses are associated with a coloring scheme to enable end users to easily detect critical tasks. For example, tasks that are not started are shown with black header, tasks in progress are shown with orange header, completed tasks receive a green header and overdue tasks a red header. Further, the description link within a task instance node opens a dialog with full task (text) description.

Tasks attachments (artifacts) that are added in CTM tasks in the personal to-do list, are externalized to a central artifact repository on the CTM server, and are accessible in the task instance nodes in the task delegation graph overview. Clicking on an attachment icon (upper right corner of "Bind EDI customer" task in Figure 8.3) opens a list with all attachments in a task instance, where each attachment can be opened separately (see *Attachments* dialog on the right in Figure 8.3).

Various further operations are enabled on task delegation graphs such as expanding and collapsing of user containers. Collapsing a user container hides the contained task hierarchies and reduces the size of the container by preserving the container with the displayed user name in the overall task delegation graph. Thus, through collapsing of user containers a kind of filtering of the overall task delegation graph is provided to display only the task hierarchies of specific users. Further operations on task delegation graphs such as text search (in task instance nodes) and zooming are provided. For navigating in task delegation graphs that cannot be fully displayed on the screen, an additional navigation dialog can be used (see *Navigation* dialog on the left in Figure 8.3) where the user can drag a gray area, representing the visible screen of the task delegation graph overview, over a minimized task delegation graph representation.

To sum up, the task delegation graph overview enables end users to recognize their position and role in overall enterprise processes, to identify potential bottlenecks, and to evaluate work distribution by viewing task instance, artifact, and user information for an evolving ad-hoc process instance. Through this the task delegation graph overview provides enhanced transparency into the evolving collaborative tasks (R3) and enables users to act as *"informed participants"* [FGY+04] in the composition of weakly-structured process models.

Through the *Show Roottasks* button in the tool strip of the task delegation graph overview (see upper part of Figure 8.3) the user can open a list with all initial process tasks (root tasks) generated on the server throughout the whole enterprise. Within this view the user can navigate through the root tasks list and open a task delegation graph for a given root task. Through this explicit exchange of process knowledge is enabled (R4).

An important notice here is that the focus of the provided implementation is set on the composition and adaptation of process models by business users, who can share information without extensive privacy requirements. Therefore no fine-grained authorization framework is currently provided. Such needs to be considered for CTM usage in a larger enterprise context.

8.5 SER of Weakly-Structured Process Models

Seeding, evolutionary growth, and reseeding (SER) [FGY+04] of weakly-structured process models is enabled in CTM through mechanisms for extraction, adaptation, exchange, and reuse of

process knowledge (R4) in the form of task patterns. These mechanisms are provided through the Task Pattern Explorer/Editor component which is shown in Figure 8.4.

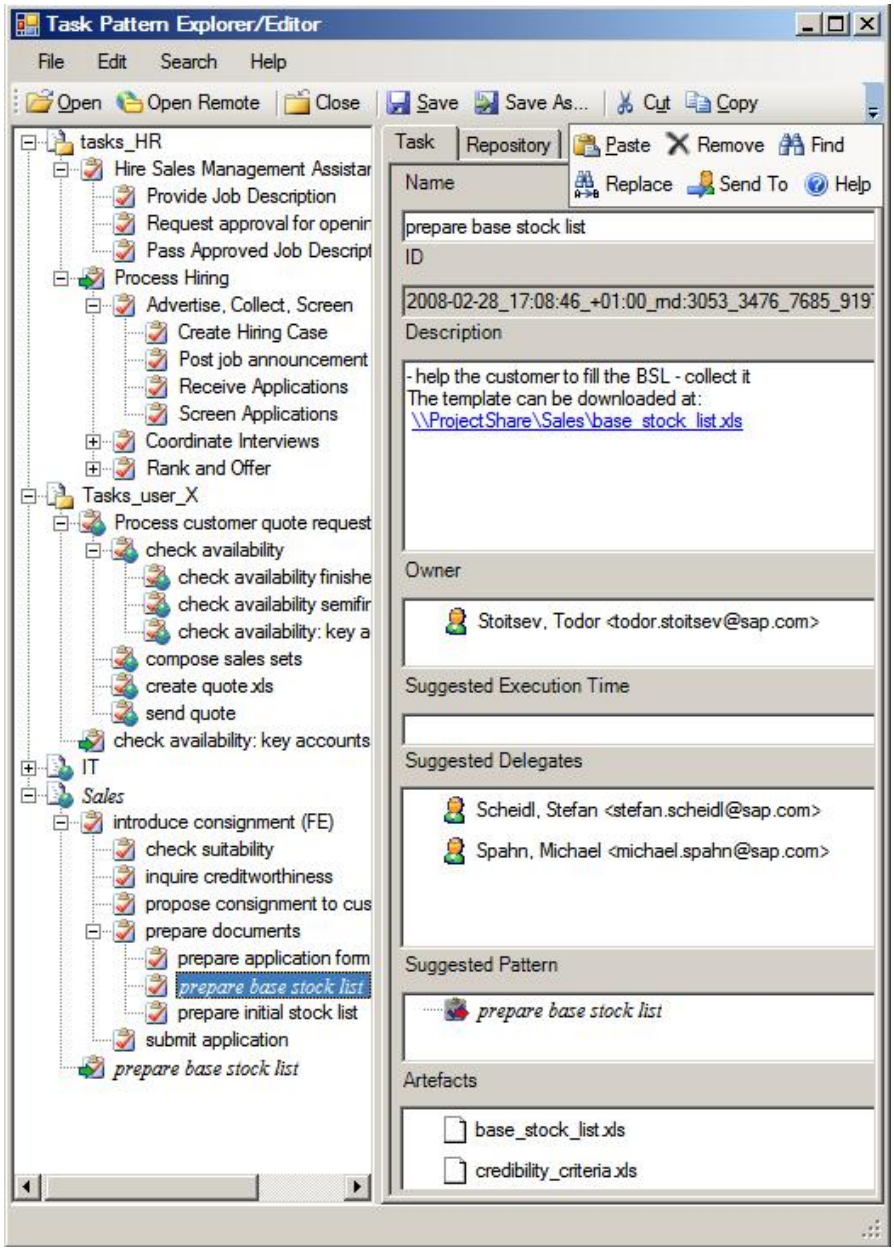


Figure 8.4: Task Pattern Explorer/Editor

8.5.1 Extraction

CTM enables extraction of a local task (including sub-tasks) from the personal to-do list to a single task pattern, as well as export of complete task delegation graph from the server to multiple task patterns, which are interlinked based on suggestions according to the delegation flow. A task pattern can be saved in a local or remote task pattern repository.

Local task pattern repositories are implemented based on the Extensible Markup Language (XML) [W3C03]. A local task pattern repository is a XML-based document, representing tasks with their complete sub-trees, context information (e.g. subject, description, owner, recipients

etc.) and references to used artifacts and suggested task patterns.

Remote task pattern repositories reside in a database on the CTM server. In the tree view on the left in Figure 8.4 a local repository node is *tasks_HR*, and *Sales* is a global repository node.

8.5.2 Adaptation

The Task Pattern Explorer provides rich editing and search functionality – all fields with white background in Figure 8.4 are editable. Cut, copy, paste, insert, and remove operations are enabled on task trees, as well as on data in context fields (on the right hand side). Task-related dialogs (cf. Section 8.3) are not extracted into a task pattern as these are considered part of the context of a concrete ad-hoc process instance. However, a relation to the original task execution and the related dialog is possible through the evolutionary relationships discussed in Section 8.6. Creation of a task pattern from scratch as explicit best-practice representation is also supported.

When editing task patterns in the Task Pattern Explorer *"the user is not required to interact in the interface domain of computational abstraction, but works directly with the data that interests him or her"* [Bla06]. In this sense CTM enables programming by direct manipulation of the task pattern fields. The *Name*, *Description* and *Suggested Execution Time* fields hold simple task context information in text format and are self-explanatory.

The *Owner* field provides expertise recommendation and represents the person, who has general expertise related to a given task. When a task pattern is extracted from an ad-hoc task instance, the owner is the person in whose to-do list a task was residing.

The *Suggested Delegates* field contains information about the persons, who have the expertise to execute a given task. When a task pattern is extracted from a collaborative process, the task recipients are set in this field.

The *Suggested Pattern* field holds a reference to a task pattern, which may be used for the further processing of a task. When a task pattern is extracted from a task delegation graph, such references in requesters' tasks point at recipients tasks, used for the further task processing. The recipient tasks are themselves extracted as separate task patterns (cf. Section 5.2.2.3). In case of task delegation to multiple recipients, in the extracted task pattern the task of the first recipient is set as suggested task pattern by default (see Figure 8.5). The generic assumption is that all recipients have executed the same activity, e.g. an annual task for performance management assessment is sent by a manager to their complete team, and all team-members basically need to perform the same procedure (cf. Section 5.2.2.3.1). Therefore one preferred task pattern can be set as suggested task pattern for future task execution for all recipients. The unused references in the *Suggested Delegates* field can be automatically removed along with the corresponding unused recipient task patterns. This removal produces a task pattern as shown in Figure 8.4.

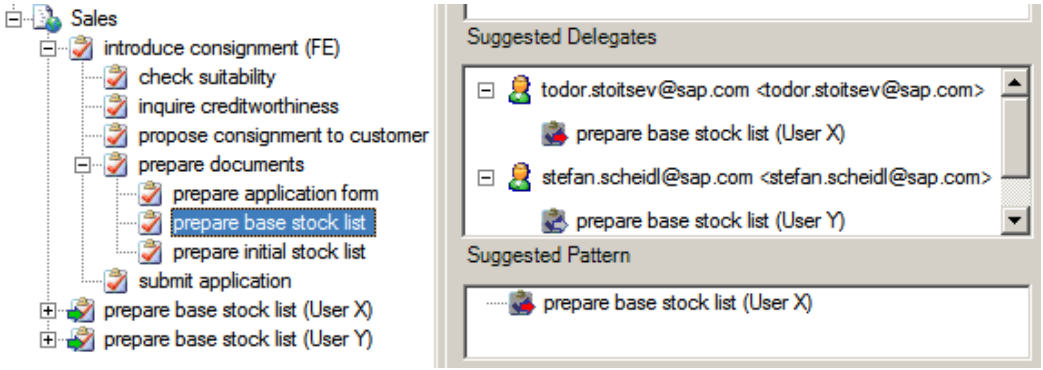


Figure 8.5: Extracted task with multiple delegations

In case of significantly different requester and recipient tasks (cf. Section 5.2.2.3.2), the requester task pattern can be refined through adding appropriate sub-tasks and matching them with suggestions to the corresponding recipient task patterns. Suggested task patterns can be added or removed in a straightforward manner - adding is done by simply copying a (root) task from the tree on the left and pasting it in the suggested task pattern field. Suggested task pattern references are supported from a local task to a task pattern in the same local task pattern repository or to a global task pattern, and from a global task to a global task pattern. Referenced global task patterns may reside in various task pattern repositories (e.g. of various departments).

Finally, attachments to tasks are represented in task patterns as *Artifacts* (cf. Figure 8.4). Custom adding of artifacts to a task replicates (externalizes) these to the artifact repository.

8.5.3 Exchange

In the Task Pattern Explorer the user can browse through different task pattern repositories and search for tasks on the server based on different criteria (owner, subject, description etc.). The results provide links to tasks in task pattern repositories and in the tracking repository. Tasks from task pattern repositories can be opened in the Task Pattern Explorer. Tasks from the tracking repository can be additionally viewed in the task delegation graph overview (cf. Figure 8.3). Through inspecting task pattern or task instance context information, the users can estimate the applicability of the respective task pattern to their current task.

No advanced proactive information delivery on tasks [HRD+06] is currently provided. CTM considers that that many users approach their colleagues for help prior to looking for solution in the available software infrastructure [RJS02]. Therefore task patterns can be exchanged through a *Send To* functionality in the Task Pattern Explorer and as attachments in task requests.

8.5.4 Reuse

Task patterns are reused through an *Apply Pattern* operation which is available on tasks in the CTM to-do list. It opens the Task Pattern Explorer where the user can search for appropriate task patterns. If the task, on which the apply pattern operation is triggered, has a suggested task pattern, the latter is automatically opened in the Task Pattern Explorer and selected for reuse.

The application of a task pattern reactivates the captured process example by generating the complete task hierarchy and filling all pre-modeled structure and content information in the to-do list. Owner information is not applied as the owner of the resulting tasks is the user, applying the task pattern. References to delegates and suggested task patterns are set in the resulting task instances to enable unfolding of the collaborative process according to the reused task pattern.

Available delegates are suggested automatically when delegation is initiated. A user can change the anticipated (example) flow by selecting or entering different recipients. The suggested task pattern references are also available in tasks. Thereby a suggestion, stored as a reference to a recipient task in the original process execution, can be used by the person, activating the task pattern, to accomplish the task themselves without further delegations. If on the other hand a delegation is issued, the recipient task contains the reference and the recipient(s) can refer to the suggested task pattern to possibly adapt and reuse it. To enable such reuse, application of a local task pattern enables iterative replication of all referenced task patterns from the XML document to a default, user-specific global repository, where these are accessible by all other users.

8.6 Task and Process Analysis in the Context of SER

For tracing evolutionary relationships between best-practices and running processes (R5) and between different best-practice definitions (R6), CTM uses the ancestor/descendant relationships (cf. Section 5.2.7). Evolutions can be viewed in the Task Evolution Explorer shown in Figure 8.6.

In Figure 8.6 the *introduce consignment* task of user *Y* (selected node) originates from a tracked ancestor task with the same name, which has been executed by user *X* (root node). The latter task has also another descendant, resulting from its reuse by user *W* (task in the bottom). User *Y* has saved a global task pattern from their execution to a remote task pattern repository (expanded node with black descendant icon under selected node). The global task pattern has been reused in two further executions. The task instance of user *U* resulted in a second global task pattern version. The task delegation graph and dialogs of tracked ancestor/descendant task instances can be shown through the *View in Repository* button for case analysis.

Initially, ancestors/descendants for a given task are provided without their sub-task hierarchy to simplify the evolution tree. The user can chose the sub-task hierarchies of which ancestors/descendants they wish to inspect. The sub-task hierarchy of a given ancestor/descendant can be retrieved through the *Fetch Full Content* button.

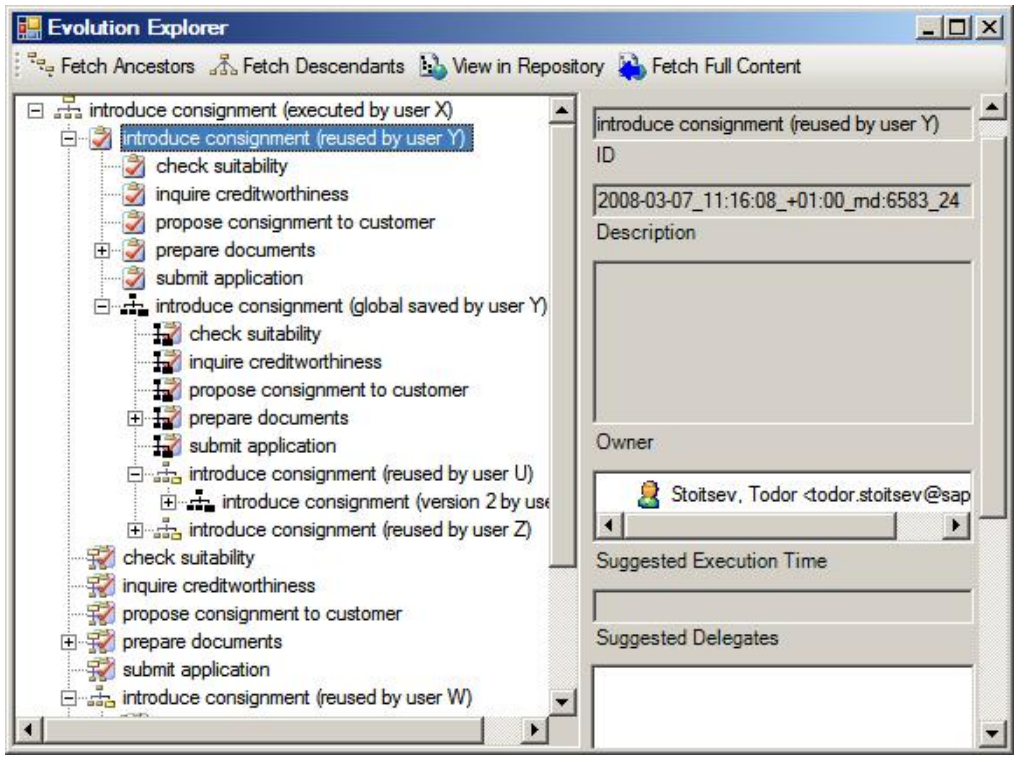


Figure 8.6: Task Evolution Explorer

8.7 From Email and To-Do to Formal Process Models

For process automation CTM uses the JBoss Business Process Management (jBPM) solution [jBPM]. jBPM workflows are modeled in a graph-oriented language – the jBPM Process Definition Language (jPDL). The workflows can be deployed and executed on a JBoss server, where they are accessed over a web front-end. jBPM processes are originally modeled in a jPDL designer, provided as an Eclipse [Ecl09] plug-in. CTM enables transformation of task delegation graphs to formal workflows in the Outlook add-in.

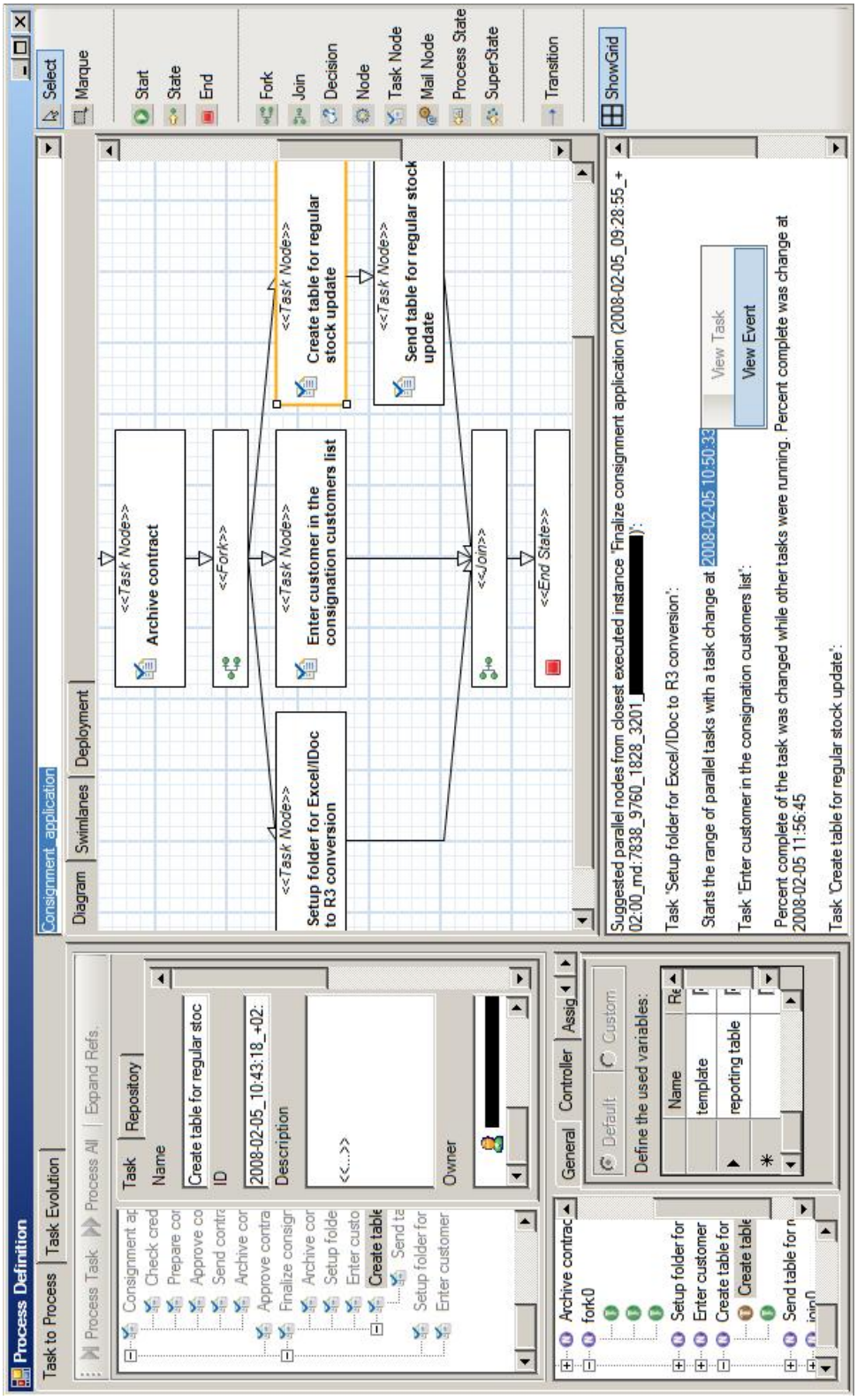


Figure 8.7: CTM Workflow Editor

8.7.1 Shared Context for Process Tailoring

CTM provides a Workflow Editor which enables a shared context between user-defined and formal process models. The Workflow Editor is shown in Figure 8.7. The upper left corner contains a view, displaying the task hierarchy in the same manner as the Task Pattern Explorer. Processed tasks receive the jBPM task icon and a gray foreground. Tasks can be processed along the hierarchy through the *Process Task* (stepwise) and *Process All* (iteration) buttons.

Transformation control is realized through various forms and dialogs. A form is provided for defining task instance change types that should be considered as task processing changes (see Figure 8.8). Task processing change types can be set before the transformation and altered between the processing steps. Further, if canceled tasks are detected during a transformation step, a dialog prompts the user if they wish to include these tasks in the transformation.

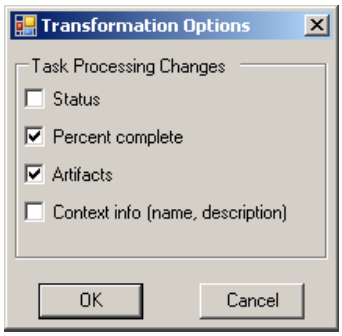


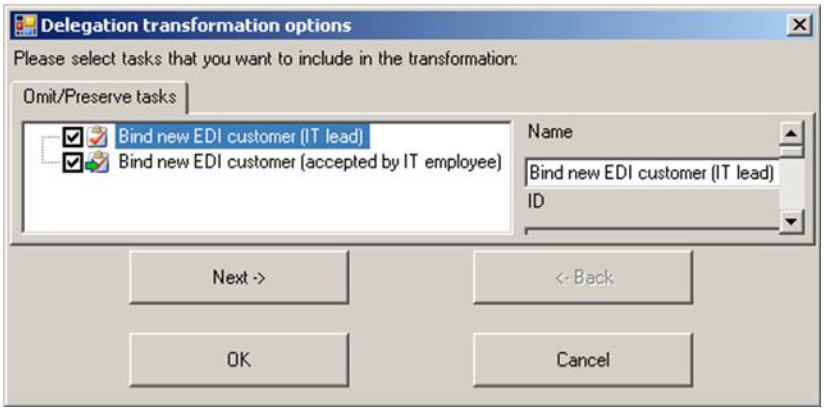
Figure 8.8: Transformation options form

Additional dialog is provided for choosing export modes for delegated tasks during process model transformation. Figure 8.9 (a) shows the dialog for preserving and omitting tasks. Tasks that are not checked in the provided tree view on the left are omitted. To facilitate the decision of which tasks to preserve or omit, task information is provided in the fields on the right hand side in Figure 8.9 (a) in the same manner as in the Task Pattern Explorer (cf. Figure 8.4). After the user has selected which tasks to preserve, they can switch to the view for merging the preserved tasks through the *Next* button. This view is displayed in Figure 8.9 (b). Omitted tasks are not included in the tree view for the merge operation. The user can copy a task from the tree view on the left hand side through a context menu and paste it in the *Merge task* field of another task on the right hand side. The constraints for merging tasks (cf. Section 6.2.4.3) are enforced during the copy and paste operations. In Figure 8.9 (b) the “Bind new EDI customer” task that has been accepted by an IT employee is set as a merge task for the “Bind new EDI customer” task of an IT lead.

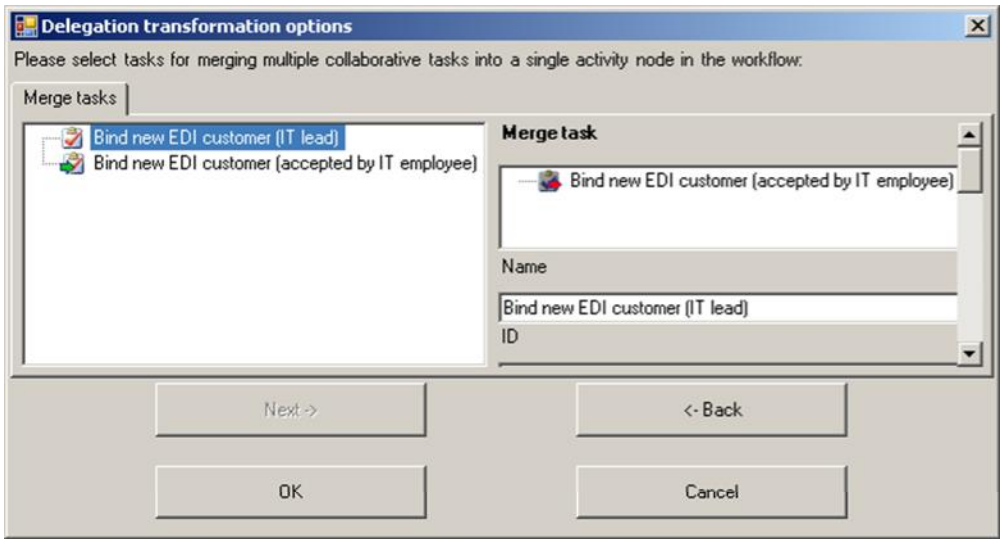
A corresponding dialog for the interpretation of hierarchical decomposition (cf. Section 6.2.3) is also provided as shown in Figure 8.10. The option for preserving a parent task as a logical group association is disabled as the current jBPM implementation does not support grouping. An option for merging a task with its parent task hierarchy is provided through a checkbox, which is only enabled if the *Omit* option is selected.

Support for resolving inconsistent task ranges through consolidation is provided in an additional dialog, which is displayed when inconsistencies are detected during task transformation. This dialog is shown in Figure 8.11. The tasks that need consolidation and the current relationships between them are displayed on the left hand side. The consolidation options are displayed in the middle of the dialog, followed by a list of consolidations that have been performed so far (in the bottom). Information about all tasks that are affected by the currently selected consolidation option and the relationships that will result after this consolidation are provided on the right hand side. Consolidation options that perform operations opposite to a

previous consolidation, are colored differently (with red foreground) to keep the user aware of the conflicts. For example, in Figure 8.11 a consolidation has been performed that adds parallelism between tasks A_j and A_q. This consolidation is reflected in the consolidation history (in the bottom). The consolidation for removing parallelism between task A_j and a further task A_k is colored in red as this split consolidation affects also tasks A_j and A_q and will remove parallelism between them (as indicated in the explanation text on the right). The consolidation environment is currently rather simplified and can be extended to integrate the complete change history of the affected tasks, and links to the task delegation graph of these tasks to facilitate the choice of consolidation options. The task change history and links to the task delegation graph are currently provided in other components of the workflow editor as discussed later on.



(a)



(b)

Figure 8.9: Dialog for selection of export modes for delegated tasks

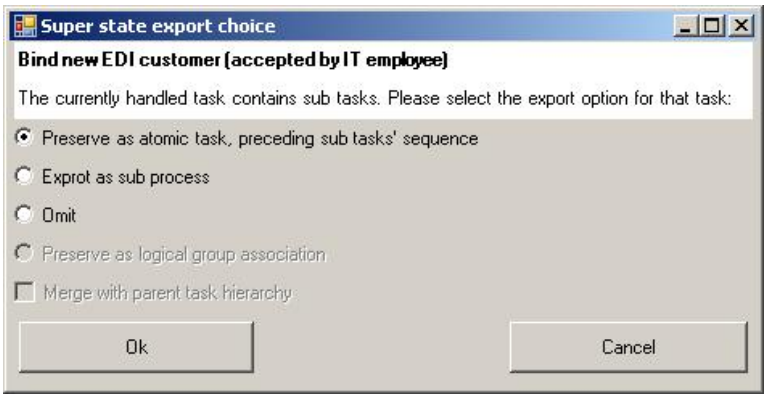


Figure 8.10: Dialog for selection of export modes for a parent task

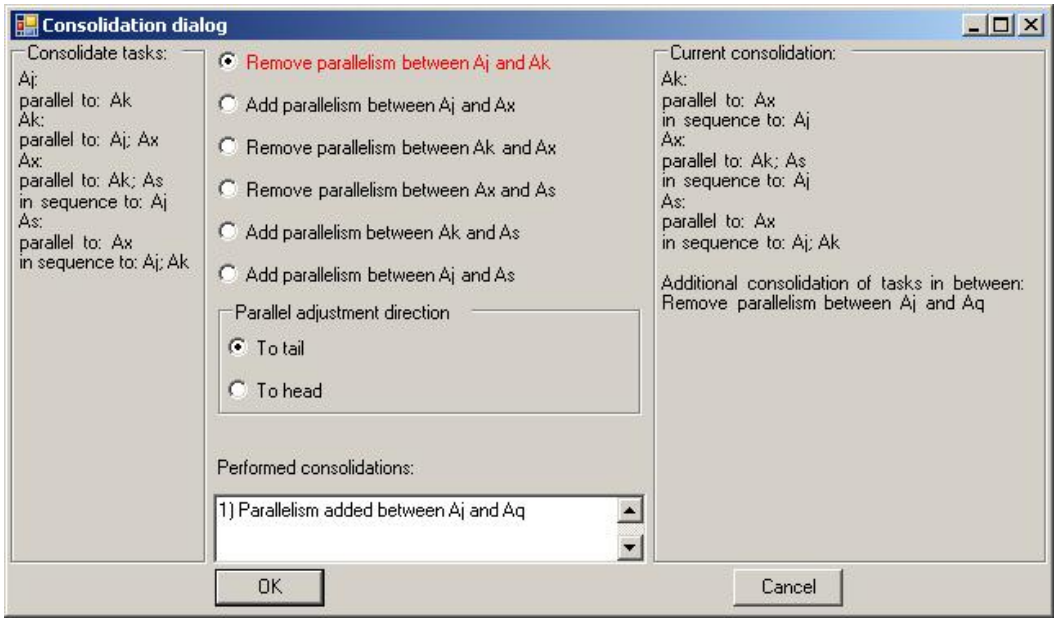


Figure 8.11: Dialog for consolidation of inconsistent task ranges

After all required transformation options have been specified a workflow block is generated and inserted in the workflow graph. The generated jPDL graph is displayed in the upper, central view of the Workflow Editor as shown in Figure 8.7.

Editing of the workflow graph is enabled through a toolbox on the right hand side in the Workflow Editor (Figure 8.7). If multiple (sub-) processes are exported, the user can switch between them in the drop-down list in the upper central part of Figure 8.7. The tree in the lower left part of the Workflow Editor contains the generated jBPM process entities (nodes and transitions). A tab control for setting their properties is provided on the right.

The *Controller* tab enables users to set parameters for task nodes. The names of available ad-hoc tasks' artifacts are set as parameters during process transformation. Users can additionally specify the artifact type – template (static) or dynamic.

An *Assignment* tab allows setting of jBPM (swim) lanes. The latter are automatically generated based on the email address of the user, in whose to-do list an ad-hoc task was residing. Each lane is defined through an expression *user(email_address)* (lanes can be edited in a dedicated *Swimlanes* tab that is visible in the upper central part of Figure 8.7). If role-based

assignments are required, the user needs to replace *email_address* with *role* in the assignment expression. The roles are accordingly maintained in the jBPM system. Thus *explicit mapping* of task assignments can be performed manually by inspecting the user data in the jBPM system.

A *Form* tab in the task properties tab control provides a text area with the code of a jBPM task's web form. The code is provided in the Extensible HyperText Markup Language (xhtml) [W3C02] as required by the jBPM system. CTM automatically generates this code by additionally embedding links to the original task delegation graphs of ad-hoc tasks, used template artifacts (available in the artifact repository), and controls for uploading dynamic artifacts and for creating ad-hoc tasks for deviations from a jBPM task instance. Advanced users can change the generated code to enhance the workflow tasks' views in the jBPM front-end.

A **transformation rationale** is provided in the lower central part of Figure 8.7 through a textual explanation of the transformations that are relevant for a selected task. The text describes the overlapping ranges and refers to the corresponding change events. A list of performed consolidations is provided in an additional dialog to facilitate the validation of a derived process model if consolidations have been performed. Further, task change and evolution history is provided in the *Task Evolution* tab in the Workflow Editor as shown in Figure 8.12. The task evolution tree in the upper left part contains on root level the task ancestors and their references resulting from delegations, followed by the currently processed task instance, and task descendants if available. The task delegation graph of tracked ancestors/descendants can be viewed through the *View in Repository* button. Task change history is displayed in the lower tree. Changes are given with their occurrence time and changed data on the right.

Generated jBPM workflows can be saved as process files or deployed on the jBPM server from the *Deployment* tab in the upper central part of the Workflow Editor (cf. Figure 8.7). Process files can be copied in the jPDL designer, where the workflow models can be extended by developers with programming code for exception and event handling.

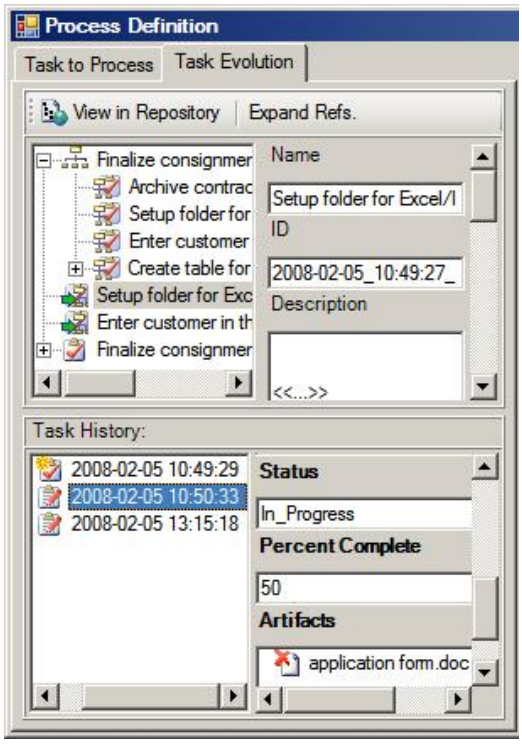


Figure 8.12: CTM Workflow Editor – task change and evolution history

8.7.2 Automation Support with Ad-Hoc Task Interrelation

jBPM workflows are started and monitored through the jBPM web front-end. CTM-generated workflow tasks contain in their web forms additional buttons for creating and accessing ad-hoc tasks. Creating of an ad-hoc task opens a web form for specifying recipient information (email address), task subject, and description. When the form is submitted, the data is sent to the CTM server along with the process instance and task instance identifiers of the deviated jBPM task instance. The server issues a *create task* event to the Office-Integrated Task Management Client of the specified recipient. This event creates a new CTM task in their to-do list.

When the ad-hoc task that is created from the deviation is tracked, the identifier of the deviated jBPM task instance is used to associate the resulting ad-hoc task instance to the deviated jBPM task instance on the server. The task delegation graph of the ad-hoc deviation task instance can be opened from the web form of the deviated jBPM task instance and vice-versa.

8.8 Summary

This chapter has described the implementation of the concepts presented in this thesis in a tool called Collaborative Task Manager (CTM). Integrated support for personal task management is provided in a light-weight to-do list (R1). The task management functionality considers the common users' working environment and provides unobtrusive support by additionally allowing task decomposition, monitoring of recipient statuses, and access to overall process information.

Email-based person-to-person communication for exchange of tasks and deliverables (R2) is further enabled. All emails for task delegation and exchange of deliverables are aggregated to dialogs, which provide transparency (R3) and structure in the task-related email-exchange.

Based on the email exchange for task delegation, individual task hierarchies of different process participants are bound to end-to-end task delegation graphs. The resulting process overview provides enhanced transparency in evolving collaborative processes (R3).

Extraction, adaptation, and reuse (R4) of weakly-structured process models is enabled through task pattern management in a Task Pattern Explorer/Editor component. Editing of task patterns is realized through direct manipulation of the task model data. Search and exchange of best-practices in the form of task patterns are further enabled.

Interconnection and structured comparison between best-practices and running processes (R5) and between different best-practices (R6) is enabled through a Task Evolution Explorer component. It provides task and process analysis in the context of SER based on ancestor/descendant relationships that result from task pattern reuse.

Transformation of weakly-structured task delegation graphs to structured workflow models for automation of rigidly recurrent processes (R7) is enabled in a Workflow Editor component. The latter provides a shared context between user-defined and formal process models. This shared context supports process tailoring as collaboration between end-users, process designers and developers. Through ad-hoc deviations from structured workflows, the users are further enabled to iteratively extend and refine derived workflow models.

CHAPTER 9: Evaluation

This chapter describes the evaluation of the concepts for end-user driven business process composition that have been elaborated in the thesis. The evaluation is based on real-life application of the Collaborative Task Manager (CTM) at industrial partner companies. The evaluation comprises three major phases: (i) a preliminary, short-term evaluation, providing first user feedback on the feasibility of the concepts for end-user driven business process composition and on the respective implementation; (ii) a long-term evaluation, focused on capturing user-defined processes; (iii) quantitative evaluation based no the Technology Acceptance Model [Dav85, Dav89], providing a combined assessment of all elaborated concepts.

9.1 Evaluation Approach

The purpose of the presented evaluation is to assess whether the provided concepts enable end users to compose business process models based on personal task management. A particular difficulty for the evaluation of the presented concepts results from their integrity, i.e. end-user driven business process composition is not enabled through a single concept or functionality, but through the combination of all introduced concepts – task management model, process composition methods, and underlying holistic concept and architecture. Therefore, an evaluation focusing on a single aspect cannot evaluate the provided concepts as a whole.

A further difficulty results from the fact that the provided concepts aim at enabling end-users without any programming or process modeling skills to participate in business process composition - an idea which is neither addressed in conventional applications for task management (to-do list) and collaboration (email), nor in business process management systems. The evaluation focuses on emerging process models that are composed by end users from scratch in an underspecified manner, whereas there are no fixed performance indicators for the respective processes and no existing formal models or workflow management systems that currently support them. Therefore, techniques such as multi-criteria analysis [vdAvH02] cannot be used to compare the CTM system with similar systems that are used for managing these business processes, in order to evaluate if CTM provides a better usability or produces more consistent and error-free process models in the complete spectrum – from weakly-structured to structured process models. Such comparative analysis is hindered also through the fact, that end-user driven process composition in CTM is an iterative process, where initial process models can be refined through repeated reuse in recurring business cases. Thus process composition by end users is a continuous process without fixed time frames. Evaluation of user-defined process models through business activity monitoring and process mining techniques [Wes07] is also not possible because there are no fixed performance indicators or strict rules for such emerging, ad-hoc processes.

To provide a comprehensive evaluation of the elaborated concepts by investigating different perspectives of end-user driven business process composition, evaluation studies consisting of two qualitative evaluation phases and a quantitative evaluation phase have been conducted.

9.1.1 Qualitative Evaluation

Related literature reports that large-scale tailoring systems are *"difficult to evaluate using objective criteria, because the complexity of the system requires complex evaluation scenarios, which makes any evaluation study difficult to repeat"* [MM00]. As a result qualitative and informal evaluation methods are commonly used for evaluation of systems where end-user tailoring takes place [MM00, NM90]. Qualitative evaluation techniques such as "think aloud" and contextual enquiry methods [BH98] are further used to complement observations based on

concrete evaluation metrics during the evaluation of end-user development systems [KM04, PBC06]. Evaluation of email-centric task management through qualitative techniques is also reported in the literature [BDHS03]. Thus qualitative evaluation has been chosen for the first and second evaluation phases as discussed later on in the dissertation.

The discussed approach for end-user driven business process composition considers end-user development in an organizational setting by using a broad technological foundation. The development and evaluation of a generic model and criteria for assessment of end-user development in organizational settings is a complex issue, which is in the focus of ongoing end-user development research [MSL06]. The latter study reveals that different application domains have different requirements, barriers, and motivating factors for end-user development. The development and validation of a model for uptake of end-user development for business process composition, including discovery and validation of generic criteria for assessment of such end-user development is a complex task that is beyond the scope of the thesis. The assessment criteria that are considered in the presented qualitative evaluations relate to the generic requirements for end-user driven business process composition that have been elicited during the empirical studies from Chapter 2. The following criteria have been considered:

- *End users should be capable of composing (weakly-structured) processes models without any or with minimal cognitive effort.* Composition of weakly-structured process models by end users enables support for ad-hoc, collaborative processes by providing structure and transparency (R3) similarly to conventional workflows. Cognitive efforts are reduced by embedding process composition in the conventional end users' applications for task management (to-do lists (R1)) and collaboration (email (R2)).
- *End users should be capable of extracting reusable task structures with different granularity, both as task models and as process models (cf. Definition 1.4 and Definition 1.5), from user-defined (weakly-structured) processes, adapting these structures, and reusing them to compose (weakly-structured) processes in recurring cases.* The extraction, adaptation and reuse of user-defined task structures address (R4).
- *End users should be capable of analyzing differences between: (i) various process/task instances resulting from a given process/task model, (ii) variations of a given process/task model that relate to the same business case; (iii) process/task model and the process/task instances resulting from its instantiation.* Analysis capabilities address (R5) and (R6).
- *End-user tailors (local developers) should be able to derive initial, formal workflow models from weakly-structured process models, incorporating process facets that are composed from various end users.* The formalization of user-defined process models by end-user tailors aims at process automation (R7) by additionally considering a shared context between user-defined and formal process models for enabling process tailoring as collaboration between end users and business technology experts (cf. Table 3.1)
- *End users should be capable of extending structured workflows with user-defined task structures at runtime, which can be used to redesign and complement initial workflow models.* This criterion relates to end-user driven process redesign for the refinement of automated workflows (R7).

The above assessment criteria are generic and do not provide concrete metrics for computing whether a given criteria has been satisfied. The criteria strongly focus on the software artifacts that are developed through end-user driven process composition, i.e. on whether end users can successfully compose process models or not. In the provided evaluation these criteria are investigated in the context of qualitative observations. Organizational factors affecting end-user driven business process composition are not considered as preliminary assessment criteria for the concepts developed in the thesis but are rather discovered through the observations of the CTM system's usage in an organizational setting. Further research is needed to develop generic

assessment criteria for end-user development for business process composition by taking into account the quality and amount of developed software artifacts, i.e. user-defined process models, but also motivational barriers, costs, benefits and risks [MSL06] for end-user driven business process composition from personal and from organizational perspectives. Some factors affecting the uptake of end-user development for business process composition on a generic level are considered in a questionnaire-based qualitative evaluation as discussed in the next section.

9.1.2 Quantitative Evaluation

The questionnaire-based empirical studies reported in Chapter 2 followed the basic idea that the uptake of end-user development increases with an increased individuals' intent to perform end-user development, because intent is a key determinant for action. This idea is fundamental for well-established information systems theories for the acceptance and use of information technology such as the Technology Acceptance Model [Dav85, Dav89]. TAM provides a quantitative method for evaluating the potential acceptance of a given technology by end users by focusing on two major aspects – perceived usefulness and perceived ease of use. Perceived usefulness and ease of use can be used to predict users' intent to use information technology as one of the major factors affecting end-user development uptake. Related literature reports TAM-based evaluation of task-centric activity support [BBPS06, RC07] which assesses the users' acceptance of the provided systems and through this the feasibility of the underlying concepts.

TAM has been developed as information systems theory from the Theory of Reasoned Action [FA75]. The latter theory postulates that voluntary behavior of individuals can be predicted through their attitude towards the behavior and the individuals' perception of how other people would view them if they perform this behavior (subjective norm). This theory is extended through the Theory of Planned Behavior [Ajz85], which introduces the concept of self-efficacy in addition to attitudes and subjective norms. Self-efficacy relates to the perceived behavioral control, i.e. it expresses the conviction of individuals that they can successfully execute the anticipated behavior. The concept of self-efficacy originates from the Social Cognitive Theory [CHH99], where it is combined with the expectations of a valued outcome from a performed behavior. The questionnaire-based survey from Chapter 2 has put forward self-efficacy estimations as a factor that affects the individual judgment and the intent of end-users to perform end-user development.

A common premise for end-user development assessments is that the individual's decisions regarding the use of a tool feature or performing the tailoring activity are driven by the balance between the cognitive costs of performing the tailoring activity and the benefits arising for the individual out of this activity [Bla02, SLM03, Sut05]. The presented survey from the empirical studies in Chapter 2 has put forward the perceived balance of benefits and drawbacks as one of the main factors impacting EUD uptake. This balance forms the "*Expectation of economic outcome*" [MSL06] which includes personal-level factors (i.e. self-improvement vs. career sidetracking) and work factors (i.e. work effectiveness vs. impact from errors) and affects the individual judgment and the intent of end uses to perform end-user development.

To provide a combined assessment of the developed concepts for end-user driven business process composition, a quantitative, questionnaire-based evaluation has been conducted. The study is based on the TAM and focuses on different aspects of end-user driven process composition through the different components of the CTM system. Through the TAM-based evaluation an assessment of the users' intent to participate in end-user driven business process composition is made, i.e. given a technological base that supports the task management model, process composition methods, the holistic concept and architecture for gradual involvement of end users in process tailoring. The questionnaire has been further extended with questions focusing on self-efficacy estimations and benefit and drawback assessments for end-user driven business process composition. Self-efficacy, benefit, and drawback expectations have been used as factors for assessing the users' intent to participate in process composition, after users have been confronted with the CTM system that implements the concepts from the dissertation. The

assessment criteria for end-user development for business process composition used in the quantitative evaluation can be thus summarized as follows:

- *Positive perceived usefulness and perceived ease-of-use (TAM) should be obtained for systems for end-user driven business process composition.* Positive TAM measures indicate that end users have positive attitude towards using a provided system for end-user driven business process composition. Such usage is a precondition of the successful uptake of end-user development for process composition.
- *Positive self-efficacy assessment*. Such positive assessment is indication of a positive users' intent to perform end-user development for business process composition, pointing at the conviction of end uses that they are able to successfully perform business process composition given a provided system.
- *Benefit expectations exceed drawback expectations.* Benefit expectations that exceed the drawback expectations can form positive attitude of end users to perform end-user development for business process composition given a provided system.

The considered criteria for assessment of end-user development for business process composition focus primarily on the users' intent to perform such end-user development. The users intent is assessed after end users are provided with the technical realization of the conceptual framework discussed in the thesis. The evaluation focuses on the CTM system that supports the introduced task model, process composition methods, and holistic concepts and architecture for gradual involvement of end users in process composition. As discussed also in the pervious section, generic criteria for assessment of end-user development for business process composition need to account for social context, benefits, drawbacks, and risks of end-user development on personal as well as on organizational level [MSL06]. The development and validation of generic models and criteria for the assessment of end-user development for business process composition are complex issues that are beyond the scope of the thesis. The next sections discuss in details the settings and findings of the different phases from the evaluation studies.

9.2 Preliminary Evaluation

Following the incremental development cycle [Gra89, LB03] a preliminary evaluation of the discussed concepts and their implementation in the CTM system has been performed [SSFM08a]. The purpose of this evaluation was to detect the general feasibility of the concepts and the respective implementation, and to uncover areas for improvement, which can be addressed in a refined conceptual framework and implementation. The evaluation focuses on the major problem areas in ad-hoc processes, which are discussed in the empirical studies, and on the requirements for end-user driven business process composition (cf. Chapter 2) which form also the assessment criteria for the qualitative evaluation. The focus is set particularly on the concepts for composition of weakly-structured process models, which are prerequisite for involving end users in business process composition. Process automation (R7) is not considered in this evaluation phase.

9.2.1 Setting and Extent of Use

The CTM evaluation was conducted in the TXTL company (cf. Chapter 2), and involved 6 users, selected for having related, collaborative tasks:

- *Chief Officer Assistant (COA):* serves as a single point of contact to the chief officer (forwards accept/reject of contract proposals); coordinates all departments.
- *Chief Sales Officer (CSO):* manages sales department, responsible e.g. for processing of special sales (consignment), credits approval, budget planning.
- *Sales Employees (SL1 & SL2):* process sales orders, make credibility checks, participate in price definition processes, assist CSO.

- *IT Department Lead (ITL):* coordinates activities of IT department, decides about acquisition of new software and hardware; manages adaptations and extensions of existing systems.
- *IT Employee (ITE):* installs software and hardware; executes business process related transactions in internal systems; maintains documentation about executed transactions; provides guidelines for transactions execution.

ITL and ITE were dealing with computers at an advanced level but did not have any process modeling or programming skills and hence match the type of end-user tailors. The other participants were typical business users. All users used Microsoft Outlook as email client. CSO, SL1, ITE and ITL also used Outlook tasks before the CTM installation.

The evaluation was initiated with a workshop in which a 60 minutes presentation on CTM was given followed by 30 minutes individual training of each user in the basic functionalities. Detailed user guides were provided to all participants. The jBPM export functionality was not included in the installations and manuals to preserve the focus on informal process support, addressing equally IT and business users. After several days, the users were visited individually, to check how they are working with the tool and to provide further instructions.

The test use was initially planned for 4 weeks. However the CTM installation on the user computers required network adaptations as well as Outlook configuration changes. Therefore only a 2 weeks trial was finally possible. Problems with character encoding schemes suspended the CTM usage by the COA for a further week.

The evaluation concluded with short video recording and transcription of the tool use, followed by a debriefing interview, in which each participant was asked to assess the basic features and rate to what extent CTM improved their ability to manage tasks in ad-hoc processes using Likert scales [Lik32] and freeform explanations. The scales used in this phase and the respective results are not discussed explicitly in the following as a detailed quantitative assessment of the different concepts and related functionalities is provided in the final evaluation phase later in this chapter.

9.2.2 Findings

Despite the initial technical difficulties and usability issues, mentioned in the following, end users found the concepts behind CTM compelling and clearly identified the high potential to structure and optimize their activities with the tool - the average overall approval rating for CTM was 4.29 (on a Likert scale of 1: Hate it, to 5: Love it). The findings from this first qualitative evaluation phase are summarized in the following.

9.2.2.1 Missing Participants

CTM was used only for task exchange among participants in the evaluation study, while other employees were addressed per email as usual. SL2 commented:

"It really makes sense if everyone is participating. I could imagine that CTM could substitute completely email in complex processes, where transparency is highly desired. […] If someone is missing, however, this breaks the chain and does not create the value I expect."

However, the industry partner refrained from distributing the CTM to further test users due to the initially encountered installation issues. Further limitations for the extensive application of CTM are discussed in the end of this chapter.

9.2.2.2 Missing Process Context

Some users suggested that root tasks should be created by senior employees, who actually trigger processes. ITE for example commented:

"I do not initiate processes, I actually execute on them. […] I always expected to get a task request from somebody [COA, CSO] who would create a root task and distribute the sub-tasks. I then would receive a task, break it down and distribute the resulting tasks to the others [Sales]."

Due to the encoding problems in the to-do list of the COA, the latter did not send requests for a week after ITE had started using CTM. This affected also the amount of tasks ITE acted on. The above issue reveals that informal process composition can be triggered along the organizational hierarchy. Thereby senior employees can drive a top-down implementation of the *"Process of Me"* [Gar06]. Other users also saw root tasks as explicit starting points of more generic business processes. Creating root tasks for already running activities did not seem reasonable to them. SL2 for example commented:

"CTM will work really nice for newly emerging initiatives where we can start tracking everything right from the beginning. […] If we use CTM to track activities on already running processes, much data will remain outside of the captured flow and we cannot expect much value."

As a result SL1 for example had created a root task for a task, which was sent by CSO per email some time ago but was not acted upon before the CTM installation. However, no root tasks were created for ongoing activities in which SL1 or SL2 were engaged before CTM was installed.

9.2.2.3 Personal Task Management (R1) and Email-Based Collaboration (R2)

Users generally reported that creating a task in the CTM to-do list does not impede their current work practice compared e.g. to dealing with email or Outlook to-do items. SL2 reported:

"A task is a task - I clearly know that I should act on it. […] Putting it in the CTM task list does not bother me. I need to think how it should be handled anyway. If I can explicitly write that down, this only helps me to clearly structure my thoughts before executing and reduces the chance to miss something."

ITE further reported, that sometimes CSO asks him to execute transactions, which he is normally not allowed to. Before the CTM installation, ITL would preserve the emails, requesting those transactions, for responsibility tracking. Receiving a CTM task for such transactions reflected this "opportunistic" behavior in the generated process example (task delegation graph) on the server and hence in the emerging process model.

Despite the clear benefits from CTM usage for visibility on time-critical activities, users stated that email cannot be replaced fully by CTM tasks. Informal enquiries outside of a concrete process would still be done over email.

User-proposed extensions: SL1 demanded extensions in the notifications handling and suggested having notifications on each change in a delegated task and its sub-tasks – structural or context change. Notifications for overdue of delegated tasks were also requested. As a further extension, users suggested summing up *percent complete* of sub-tasks and increasing the percentage of a parent task.

Some of the users proposed that the collaborative flow on tasks should be structured better to facilitate the handling of CTM emails. The *Move CTMs* functionality (cf. Section 8.3) was not accepted well - users preferred to get CTM request messages in a dedicated *CTM Mail/Requests* email folder and responses in a *Responses* folder.

9.2.2.4 Transparency (R3)

Users highly approved the overview functionality provided by task delegation graphs and dialogs and the provided notifications on task changes as they saw in them the potential to reduce overhead for calling colleagues and writing emails with task status enquires. SL1 reported:

"Such processes [price definition] draw like a red thread through the whole company. I certainly want to know how far things have gone. […] It is annoying when you do not get feedback on requested actions. This [CTM process overview] will save me the effort to constantly call people or write mails to ask about the status of things."

Generally, employees with managerial functions had greater interest in the overview functionality than others. SL2 for example stated that seeing what others do might not be of interest to him as it might concern activities outside of his expertise scope. COA, CSO, SL1 (who had more senior functions) and ITL clearly wanted an overview.

The ability to represent artifacts in process steps was also considered crucial. Different artifact versions were attached to consequent tasks in a process flow, which revealed how artifacts are elaborated within a process. For example an empty, preformatted Microsoft Excel table was attached in a request issued from CSO to SL2, and a filled Excel table was available in the resulting SL2 recipient task, which was elaborated to 75%.

User-proposed extensions: As CTM was used only by a small group of people, privacy issues were not raised during the trial. However ITL stated that authorization has to be considered for extended CTM use in the enterprise by providing the possibility to hide certain process fragments in black-box containers in the web-based task delegation graph overview.

9.2.2.5 Exchange, Adaptation and Reuse of Process Knowledge (R4)

An important perceived benefit from CTM was that the task delegation graph overview provided the possibility to fill the communication gaps between different departments. Task delegation graphs thus enabled implicit exchange of process knowledge on evolving collaborative processes. ITE commented:

"People from different departments here have their specific work practices and ways to manage information. Few know what is going on and how things are processed in other units. This tool [CTM] allows us [IT, Sales] to bridge these boundaries by streamlining the information flow or at least providing common basis for information exchange."

Users further appreciated having an example of how a problem should be approached. SL2 commented:

"Basically I have to achieve certain output for the tasks I receive [from CSO]. I really appreciate to know how she [CSO] would break down the task and what the different facets in the task are. This helps me to stay on the right track and to know what is expected of me."

However, the observations showed that actually CSO would send a single task request with generic description, e.g. *"prepare contracts for customers C1, C2, and C3"*. SL2 would then decompose this task by creating a sub-task for each customer. Therewith tasks disperse and refine by falling through the organizational hierarchy. This reveals that seeding, evolutionary growth, and reseeding (SER) [FGY+04] towards complementing abstract process definitions can happen during task execution.

ITL, on the other hand did not think that he would benefit much from external knowledge. ITL however appreciated being able to distribute knowledge himself, i.e. as a global task pattern, to

avoid repeated inquiries from other employees on same topics.

Although only several task patterns were extracted – 2 in IT department (1 global and 1 local task pattern) and 3 in sales (1 global and 2 local task patterns), the benefit from structuring process knowledge in a way that it could be reused was stated as a clear benefit. However, users were uncertain about the reuse potential of task patterns and the way these should be distributed to others. The overall attitude was that global task patterns should be delivered by a (senior) domain expert, who can handle also the responsibility for providing them. CSO e.g. experimented and developed a global task pattern instead of writing a text-based guideline. SL2 on the other hand refrained from submitting a task pattern on a remote repository while stating that he could send the local task pattern to a colleague personally, upon request, and furthermore, that he *"silently agrees"* for other colleagues to take and adapt his implicitly generated task example from the tracking repository on their own responsibility.

User-proposed extensions: A significant pain-point with respect to task patterns was that in the company best-practices were described in text documents, e.g. in Microsoft Word. Defined task patterns were experimental and the users did not spend much effort on detailing them, because they would be useful only to the small group of people that participated in the evaluation. Thus users proposed enabling conversion of task patterns to text documents which can be used also by other persons that do not have CTM installed.

9.2.2.6 Interconnecting Best-Practices and Running Processes (R5)

The users considered that comparison of task patterns and running tasks, resulting from their application, might not scale for large processes. Best-practices were generally desired as higher-level process descriptions, while running processes could produce multiple fine-grained tasks. CSO for example commented:

"As far as I am concerned a task pattern will contain only top-level tasks as my employees always do things differently. This doesn't bother me if the results are delivered on time. […] It is good to have a guideline, even if you do not care how the described tasks are accomplished concretely."

The overview provided in the Task Evolution Explorer was not considered intuitive. Differences in task structures could be identified through additional effort, which would bring benefit only to managerial employees.

User proposed extensions: Users suggested enabling task comparison in *"swimming lane"* overview, where the corresponding top-level tasks can be put against each other. This would enable users to better see the corresponding and missing process facets, by possibly discarding low level tasks. Filtering based on different criteria like e.g. task level and owner was suggested.

9.2.2.7 Tracing of Evolving Best-Practices (R6)

Despite the deficiencies in the usability of the Task Evolution Explorer, the functionality that it provided was considered necessary by senior employees because of the frequent changes in informal process guidelines. Tracing of such changes could help to at least undo wrong strategies. SL1 commented:

"We often change processes to check if we can achieve better results. We check e.g. for the processing of these contracts we needed that much time, while we have planned that much. […] If we see that a change does not deliver better results, we switch back to our previous practice. […] An overview and comparison of the tasks for both practices in CTM is nice to have."

The structural overview provided in the Task Evolution Explorer was considered still

insufficient as users cared also about certain performance indicators.

User-proposed extensions: Users proposed that the comparison of task hierarchies in the Task Evolution Explorer should be enabled based on specific criteria like e.g. execution time or persons involved. It was further suggested that in addition to the ancestor/descendant relationships also versioning of task patterns should be supported.

9.2.3 Summary of Findings

The findings from the preliminary evaluation phase show that the developed concepts are feasible for supporting end-user driven business process composition in the context of ad-hoc processes. An important observation for ad-hoc process composition is the top-down realization of work distribution, where domain experts and managerial employees trigger processes by declaring, (eventually decomposing) and delegating tasks. Thus a top-down implementation of the *"Process of Me"* [Gar06] is provided where SER of weakly-structured process models is supported through: (i) refinement of generic tasks during execution; (ii) adaptation and reuse of process fragments (task patterns). Opportunistic and emergent changes [WJ04] in running informal processes are supported through ad-hoc task decomposition and delegation. Such changes in best-practices and enterprise guidelines are supported through allowing underspecified, high-level task patterns, which can be ascertained during an actual process execution and then extracted as "evolved" best-practices.

The preliminary evaluation further delivered user-proposed extensions. These have been partially considered and implemented in a second prototype version as discussed in the following.

9.3 Long-Term Evaluation

After the preliminary evaluation some of the user-proposed extensions were implemented to increase the acceptance of the solution. For example, export of task patterns to Microsoft Word documents was implemented. An exported task pattern document contains a numbered, hierarchical list with the tasks (task names come as titles in the numbered list), description is provided under each task, and artifacts are retrieved from the server, copied to the target export folder and referenced through hyperlinks in the document. Not all user-proposed extensions could be implemented due to the limited time frame between the first and the second evaluation phase. The second phase consisted of a long-term application of the CTM prototype in real-life settings. The gathered data was analyzed and subsequent case studies [SSFM08c, SSFM08d, SS08] were performed to evaluate the concepts related to composition of structured process models.

9.3.1 Setting and Extent of Use

The 6 users from the preliminary evaluation phase were recruited for the long-term evaluation. The long-term evaluation was initiated with a "refreshment" workshop, comprising a 1 hour presentation on CTM, followed by 30 minutes individual training of each user on the basic functionalities, to ensure that the users are familiar with the tool functionalities. Detailed, updated user guides were provided to all participants, containing also the additional functionalities that were implemented for the second prototype version. The jBPM export functionality was again not included in the installations and manuals to preserve the focus on informal process support, addressing equally IT-skilled and business users.

The long-term trial lasted 8 weeks. Daily backups of the CTM database were scheduled and collected for evaluation each week. After the first week of test usage the users were visited individually, to check how they are working with the tool and to provide further instructions. The evaluation phase concluded with a short video recording and transcription of the tool use, followed by semi-structured debriefing interviews.

Based on the database excerpts and interviews, a set of captured ad-hoc processes was selected for transformation to structured workflows. Case studies were conducted in subsequent iterations within this evaluation phase to evaluate the concepts for derivation of structured process models. These case studies are detailed in the following sections.

9.3.2 Findings – Composition of Weakly-Structured Process Models

The findings from the long-term evaluation regarding ad-hoc process support generally confirmed those form the preliminary evaluation phase. Additional aspects are discussed in the following. An excerpt from the long-term evaluation metrics is given in Table 9.1. The given data does not include the process instances and task patterns from the short-term evaluation.

Table 9.1: Evaluation metrics

Metric	**N**
Created root tasks (ad-hoc processes)	8
Created tasks (overall number of task instances without root tasks)	46
Delegations	14
Unique attachments added	25
Attachment changes (diff. checksum, same name)	12
Percent complete changes	45
Task changes overall (only edit, no create/delete)	68
Created remote task patterns	2
Created local task patterns (files on user PCs)	4
Reused remote task patterns	1
Reused local task patterns	2

9.3.2.1 Process Structure and Level of Specificity

A highly varying number of tasks per process instance were detected (mean 6.75, std. dev. 4.03, median 5.5, min 3, max 15). Two processes contained only 3 tasks and one delegation, thus representing exchange and coordination of high-level work items between two persons. However, one of these processes reoccurred two more times, respectively with 5 and 7 tasks. The increasing number of tasks implies that a need to detail the process has been recognized with its reoccurrence. This process was for binding a new customer for Electronic Data Interchange (EDI). It is discussed in more details in a process formalization case study later on in this chapter.

Two of the captured ad-hoc processes which were elaborated at a high level of details (one with 10 tasks and the other with 15) were related to the consignation sales process, which was discussed during the empirical user studies (cf. Chapter 2 and Appendix B), and used as introductory scenario for CTM usage in the company. Thus the users were focused on this process and considered streamlining it in a long-term perspective.

9.3.2.2 Personal Task Management – Managing Task Context Information

The observation from the preliminary evaluation that CTM does not impede the users' work practice related to personal task management was confirmed in the long-term evaluation. An additional observation was that users generally manage percent complete and status information, however not as precise estimation of work completion, but rather as a rough progress indication. ITE for example reported that he managed status merely to indicate that he is working on a given task and to avoid getting calls and emails from sales.

Users further maintained attachments in CTM tasks. Some users considered this as a convenient way to share documents with their colleagues without a manual effort for repeatedly sending emails or copying documents in a shared folder. SL1 for example reported that document

exchange through CTM tasks was faster than email as she attached an updated document to a task and the other users could pull the latest version from the process overview.

9.3.2.3 Exchange, Adaptation and Reuse of Process Knowledge

The observation from the short-term evaluation that task patterns are considered useful was confirmed. However, due to the restricted CTM usage, it was still not possible to distribute task patterns throughout the company. This prevented from developing a global strategy for task pattern management as alternative to text-based documents.

The Microsoft Word export functionality was not used, as users considered that if they need a global guideline they would start writing it in a text document. Creating it as a task pattern and exporting it to Word would require additional effort for formatting the document according to their needs later on. Task pattern export to textual documents was considered useful, if task patterns are *"the rule"* for defining guidelines, and text documents are a *"workaround"* which is needed in exceptional cases. Currently, the case was exactly the opposite – only a small group of users could use task patterns whereas all other employees in the company used text documents.

Eventually, only 2 global task patterns were available (from ITL and CSO) whereas SL2 and ITE had developed local task patterns. All task patterns were experimental and intended for use only by the group of test users.

9.3.2.4 Observed Discrepancies and Required Concept Extensions

While task delegation graphs are suitable for top-down planning and execution of ad-hoc processes, issues were detected when processes emerged in a bottom-up manner. For example, in a captured ad-hoc process for settlement of consignation sales (for details see Section 9.3.3.3, Figure 9.4 (a)), SL2 had delegated a task *completeness feedback* to accounting (CSO), and after checking the completeness feedback, CSO had to *check the customer payment*. The overall process was initiated by SL1. In a correct top-down implementation where SL1 coordinates the whole process, the *check payment* task would be defined by SL1 and delegated to CSO. In this case, after CSO completes the *completeness feedback* task, the corresponding requester task of SL2 would also be completed (after approval of the completion declaration). Thus, the *check payment* task of CSO would remain the only opened task in the process as it depends on a customer action, whereas SL2 has finished his work in the process.

In the captured process, SL1 had not created and delegated a *check payment* task to CSO. Instead CSO had created a *check payment* task herself. Creating this task as a new root task would decouple it from the current process and produce another process. Thus, CSO had created this task as a sub-task of the accepted *completeness feedback* task, which was delegated by SL2. The hierarchical order did not allow completing the *completeness feedback* task of CSO before the *check payment* task has been completed (completion of parent task completes also sub-tasks). As a result also the requester task *completeness feedback* of SL2 could not be completed before the customer payment, although SL2 has finished his work.

Users commented that on the one hand, it may be useful to keep all stakeholders involved in an ad-hoc process until a final (external) action confirms that the final goal of an ad-hoc process has been reached (i.e. customer payment). On the other hand, however if the customer delays the payment, the *completeness feedback* task of SL2 will be marked as overdue and can result in escalation, although SL2 has finished his job. Thus, concept extensions are needed to enable adding of unplanned tasks (e.g. *check payment*) to an ad-hoc process in a bottom-up manner. To deal with the discussed discrepancy the thesis introduces the concept of a *disjoint ad-hoc task*.

9.3.2.5 Disjoint Task – a Concept for Bottom-Up Extension of Task Delegation Graphs

A **disjoint ad-hoc task** is a task which is associated to a given ad-hoc process instance (root task), resides on root level in a users' to-do list but is not a root task or accepted task. A disjoint

task thus does not have a parent task but only a root task association and has a different owner than the root task. Root tasks and accepted tasks have been discussed in the task management model (cf. Section 4.4.1.4). A new task type *disjoint* is considered for disjoint tasks.

A disjoint task for a given ad-hoc process can be created by a user, only if there is an existing accepted task for that process in the users' local to-do list. This is needed to enable association of a disjoint task to an ad-hoc process instance from the local user workspace. When a disjoint task is created, the user is given the possibility to choose to the process instance (i.e. to the root task) of which accepted task the disjoint task should be associated. Thus, a disjoint task enables recipients to define further tasks that they need to do in the process and which have not been defined by a requester or by the person, coordinating the process. A disjoint task like any other task instance can have an arbitrary number of sub-tasks.

During task pattern extraction, disjoint tasks are extracted as separate task patterns. In contrast to delegated tasks, they are not referenced through suggested task patterns. A visual indication may be provided for extracted disjoint tasks, which informs the user that these tasks need to be arranged in the overall extracted task pattern structure.

During process formalization existing disjoint tasks can be proposed for handling as sub-tasks of the root task or together with recipients' tasks of the owner of the disjoint task, i.e. when the respective delegated requester task is handled. The first case considers that disjoint tasks would be planned as sub-tasks of the root tasks and delegated in a consistent top-down implementation. The second case considers that disjoint tasks are tasks that need to be performed prior to, or after a delegated task, but have not been defined and delegated by a process coordinator in a top-down manner. In the latter case, the rules for merging and excluding delegated tasks apply also for disjoint tasks (cf. Section 6.2.4). Once a disjoint task is transformed, it is not proposed for transformation any more.

9.3.2.6 Summary of Findings – Ad-Hoc Process Composition

The long-term evaluation results show that the presented approach for involving end users in business process composition through enhanced personal task management is adequate and efficiently reduces the cognitive distance between work tasks and end-user development (modeling) tasks. The primary perceived benefits for task management are the transparency in collaborative activities and the reuse of previous experience.

During the case study users were able to develop several weakly-structured process models. The results show that users intuitively compose ad-hoc processes in an underspecified manner. Detailed process descriptions are developed only when an explicit need to streamline a business process in long-term perspective is perceived. Process reoccurrence is a stimulus for detailing emerging process definitions and streamlining processes.

While a top-down planning and execution of ad-hoc business processes is sufficiently supported through ad-hoc, hierarchical to-do lists and task delegations, the need to enable bottom-up extensions of emerging processes has been additionally perceived. To address this need, the thesis introduces the concept of a disjoint ad-hoc task. Due to the limited timeline of the research project, disjoint-tasks could not be explicitly validated. The feasibility of the concept is based on the observed user-defined process models in the presented long-term evaluation.

During the long term evaluation users were further able to develop several experimental local and global task patterns. Although reuse of process knowledge through task patterns was considered as beneficial, a strategy for task pattern management could not be developed during the test usage. Limitations of the evaluation are discussed in details in the end of the chapter.

9.3.3 Findings – Composition and Refinement of Structured Workflows

Formalization of user-defined process models was evaluated through three different case studies. The case studies are based on the collected ad-hoc process data from the long-term evaluation.

The first cases study [SSFM08c] focuses on process tailoring by end-user tailors (local developers). The second case study [SS08a] explores process composition through involvement of end users without IT-expertise. The third case study explores extension of a formal workflow by end users through ad-hoc task deviations [SSFM08d].

9.3.3.1 Process Formalization by Local Developers

ITL and ITE were considered as matching the type of local developers (end-user tailors) as they were dealing with computers at an advanced level but did not have any process modeling or programming skills. The binding of a new customer for Electronic Data Interchange (EDI) occurred 3 times in the collected database backups. During the interviews after the long-term CTM usage, ITL and ITE confirmed that this process could be suitable for automation as it reappears basically in the same manner for various customers.

9.3.3.1.1 Setting

In a subsequent iteration after the long-term CTM usage, ITL and ITE were asked to perform formal modeling exercise for a captured instance of the EDI process. As preparation for the exercise ITL and ITE were given a 40 minutes tutorial on the Integrated Development Interface (IDE) for jBPM process modeling in Eclipse [Ecl09], and a 30 minutes tutorial to the CTM workflow transformation environment. Then ITL and ITE were asked to model the process in each of the two environments, by using a captured task delegation graph of the EDI process for the workflow model derivation and modeling in CTM. Think-aloud and contextual inquiry methods [BH98] were used to track their strategies and intents. The exercises were videotaped and transcribed for analysis. As the focus was on process modeling as result from systematic interactions in CTM rather than on modeling with the jPDL visual notation, cognitive dimensions [Gre89] of the jPDL modeling notation were not considered.

9.3.3.1.2 Process Description

A diagrammatic representation of a captured task delegation graph for the discussed EDI process is shown in Figure 9.1 (a) (screenshots are available in [SSFM08c]). The task names are freely translated by the author from German for all discussed processes. The evaluation discussed in the following is based on the most detailed task delegation graph for the EDI process, containing 7 tasks. Two further, incomplete process versions were captured in the tracking repository – one with 3 and one with 5 tasks (cf. Section 9.3.2.1).

The binding of a new EDI customer is initiated by ITL, who receives a customer visit report from a field employee. The report includes the types of EDI messages which the customer wants to exchange. ITL sends a *bind EDI customer* task with the attached report to ITE, who asks SL2 to *maintain the customer master data* in the SAP R/3 system, and starts to *setup the customer on the EDI-converter* by creating the necessary EDI message structure. When SL2 is ready, ITE *maintains the partner agreements* by mapping internal SAP R/3 message types to the EDI message types for external communication. ITE finally *contacts the customer* to initiate the EDI.

9.3.3.1.3 Findings

When **ITL** first modeled the process in the jBPM IDE, he ordered all tasks sequentially. Task names given during the process modeling by both – ITL and later on by ITE slightly differed from each other and from the captured real-life process but had the same meaning. Although ITL found drawing the task nodes and connecting them with transitions straightforward, he considered the environment very technical:

"If you show this to him [SL2] he'll probably give up the CTM trial [laughing]."

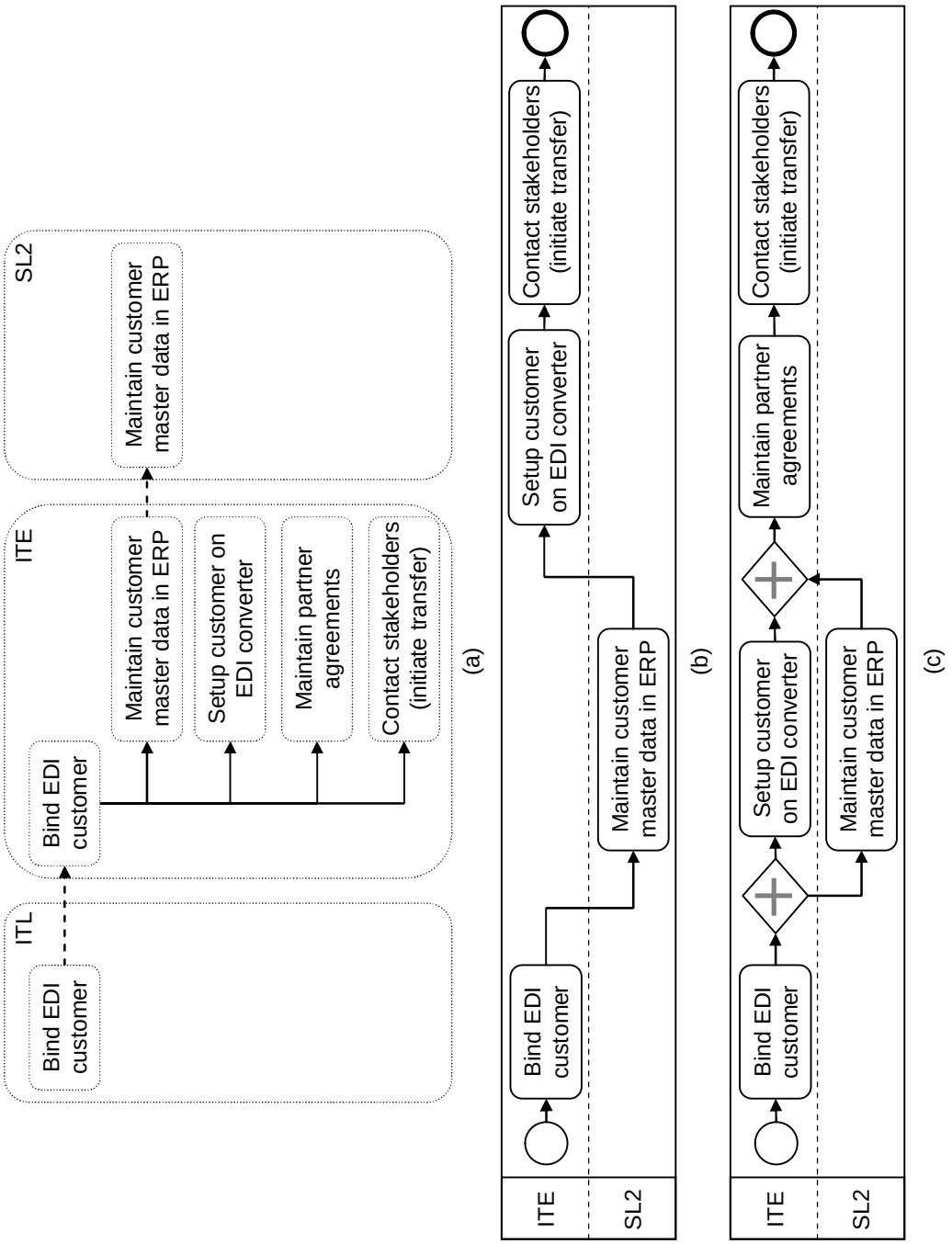


Figure 9.1: Process for binding a new EDI customer: (a) captured task delegation graph; (b) explicitly modeled process by ITL; (c) explicitly modeled process by ITE and derived process model through transformation.

Furthermore, while modeling in the jBPM IDE, ITL omitted the *maintain partner agreements* task. A diagrammatic representation of the resulting explicitly modeled workflow of ITL is provided in figure 9.1 (b) (all depicted process diagrams are based on BPMN [OMG06]).

In the next exercise, ITL was able to correctly perform the process formalization in CTM, by evaluating the generated flow through the explanation and the captured task delegation graph. Artifact changes and changes to percent complete and status were selected as task processing changes during the transformation. A diagrammatic representation of the resulting process is shown in Figure 9.1 (c). ITL commented:

"Ah, I didn't think that they do it in parallel [customer master data & EDI converter setup] … but yes, both things are independent."

Regarding the omitted task, ITL commented:

"Yes, I know that but it didn't come to my mind … he [ITE] is our expert on the topic… but here [CTM] they [ITE & SL2] have done the fine work for me, right … I need at most to cross-check with them."

When **ITE** modeled the process in the jBPM IDE, he was able to create a complete diagram by adding also parallel flow. Later on ITE performed the model transformations in CTM successfully. Thus both workflow models were consistent and corresponded to the model shown in Figure 9.1 (c). ITE commented:

"I always liked the other overview [task delegation graph], but this [jBPM graph] I like even better … they are complementary as the old [task delegation graph] gives the logical work breakdown and this [jBPM] shows you how things actually happened."

ITE also appreciated the fact that common business users like SL2 can be involved in the modeling of the *"flow diagrams"* without doing more than managing their CTM tasks. ITE commented:

"Yes, it can happen that someone misses to maintain their percent or status … but errors are OK, they will focus our attention and help us understand how work is managed or why not."

During process model transformation both ITL and ITE exported the delegated task *bind EDI customer* with omission of the requester task, as they considered that the work is actually done by ITE. The recipient task (*bind EDI customer* of ITE) was transformed to a jBPM start node. No inconsistent ad-hoc task ranges were detected during the transformation. On the other hand, during process modeling in the jBPM IDE, start nodes were specified by both participants as the trigger for the overall process. Thus in the depicted process diagrams in Figure 9.1 the BPMN start event and the first task node *bind EDI customer* translate into a single jBPM start node. A screenshot of the derived process model is available in [SSFM08c].

ITL developed the model in the jBPM IDE for approximately 23 minutes, whereas the formalization in CTM took him about 9 minutes (including evaluation of correctness). ITE needed approximately 18 minutes for modeling in the IDE and about 7 minutes in CTM. An important notice here is that the modeling in the IDE was performed with the jPDL 3.2.2 version, which requires explicit generation of xhtml web forms for jBPM tasks. This is not the case with the jPDL version 3.0. The latter version however does not allow editing the code of a jBPM task web form and thus hinders the customization of the workflow task instance view.

9.3.3.1.4 Summary of Findings

The observations showed that modeling in the jBPM IDE demanded a lot of time alone for thinking of how the process is executed and for writing the task names. Time consuming were also the setting of assignments (swimlanes) and the generation of the task forms, which are automated in CTM. The workflow developed by ITL in the IDE was furthermore inconsistent due to the omitted task and the strictly sequential flow. CTM delivered a real-life compliant process by only requiring comparison with an implicitly generated task delegation graph and selection of export mode options.

9.3.3.2 Process Formalization Through End-Users' Involvement

A process for initiation of consignation sales appeared once in the database extracts. The process was applicable to multiple customers and a need had been recognized in TXTL to streamline this process in long-term perspective. It was thus considered as a suitable candidate for formalization.

9.3.3.2.1 Setting

After the long-term CTM test usage, a follow-up workshop was organized with all test users to explore how the captured process for introduction of consignment sales can be formalized and possibly automated through a jBPM workflow. The workshop consisted of two phases.

The first phase started with a 1 hour tutorial on jBPM including jPDL modeling and execution of a sample process. A group discussion [BD06] followed in which the process for initiation of consignation sales was modeled in the jPDL visual designer by asking the users to define the tasks, their sequence and assignments.

The second phase started with 1 hour tutorial on the CTM functionality for generation of a jBPM workflow from a task delegation graph. Then a jBPM workflow was generated from the consignation sales process captured in the CTM tracking repository whereby the users were asked to select the export mode options for interpretation of hierarchical decomposition and delegation flow. Finally, the generated jBPM workflow was deployed and the users were asked to perform a test run. Think-aloud and contextual enquiry methods [BH98] were used to track their strategies.

9.3.3.2.2 Process Description

An overview of the process for initiation of consignation sales as described by the end users in the first phase of the follow-up workshop is shown in Figure 9.2. BPMN [OMG06] is used to show the document flow and stakeholders' departments. The focus of the evaluation is set on the company-internal process (*Company* pool), which does not include customer's or Field Employee's (FE) tasks. Such are shown in Figure 9.2 for completeness. The process is as follows.

A FE *checks the suitability* of a customer for consignation sales based on various criteria. Then FE *enquires the creditworthiness* of an appropriate customer from sales (generally CSO). Upon a positive response from sales, FE *proposes consignation* to the customer. If the latter accepts the proposal, FE *prepares the documents* for the application processing, i.e. *creditworthiness data*, consignation *application form* and a *base stock list*, and sends them to sales (CSO). The latter *prepare the contract* and *request approval* from the management (COA is single point of contact). Upon approval, sales (CSO) *send the contract* to the customer, who *signs* and *returns* it. Then sales (CSO) *archive the contract* and *enter the customer in a consignation customers list* (Microsoft Excel file) which is used to monitor the contract status of consignation customers. Independently from sales, IT (ITE) *sets up a folder for automated Excel/IDoc to R/3 conversion*, where regular customer sales reports are copied and automatically read into the SAP R/3 system. Sales (SL1) create a *table for regular stock reporting* and *send it* to the customer.

The company-internal task sequence from Figure 9.2 corresponds to the jPDL workflow, designed in the first workshop phase. The designed workflow contained a jBPM start node named *"New consignation customer"* instead of the BPMN start event shown in Figure 9.2.

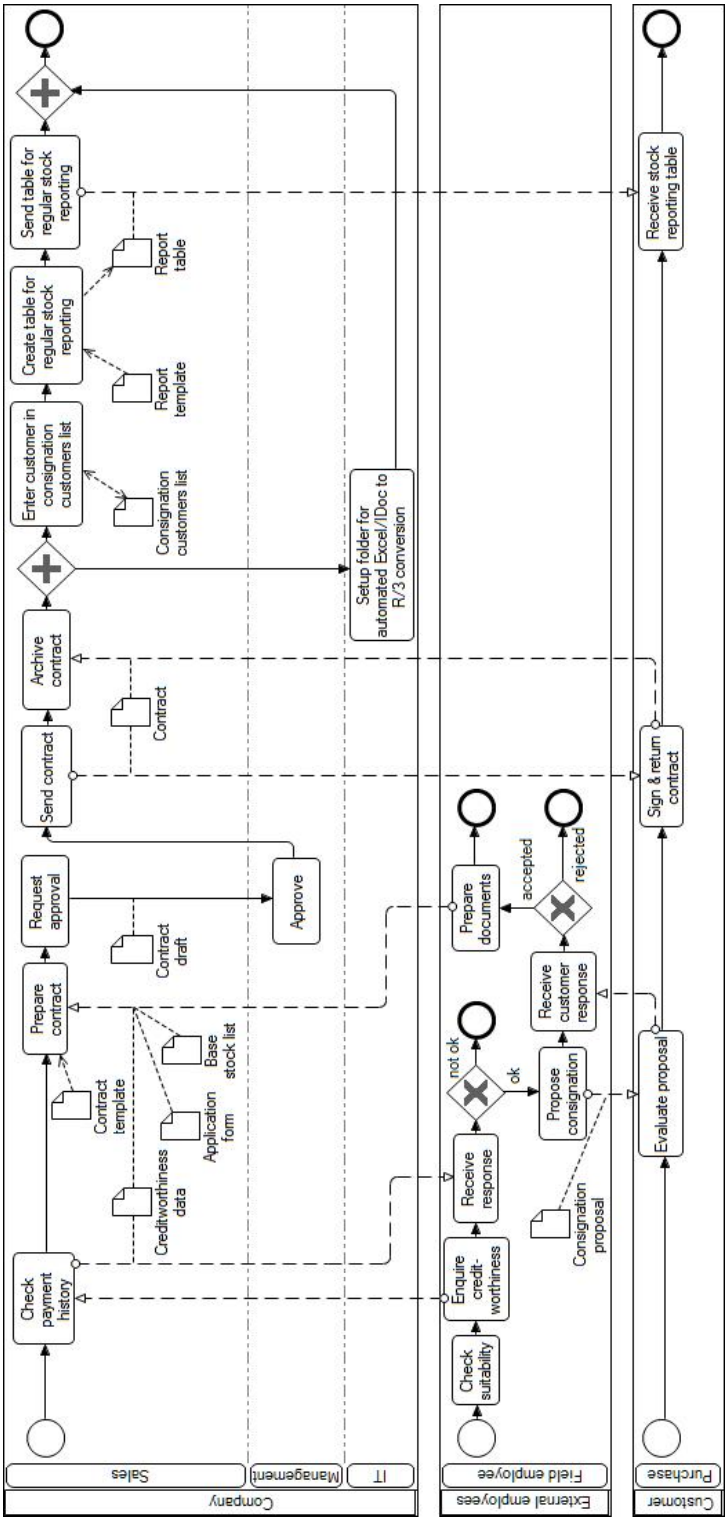


Figure 9.2: Process for initiation of consignation sales – process overview with document associations

Looking back at the Hierarchical Task Analysis (HTA) diagrams for the consignation process from the preliminary empirical studies (cf. Appendix B), differences can be found between the process described during HTA analysis and the process described in the process modeling workshop. One major difference is the procedure for automated reading of customer tables for regular stock reporting. This procedure had been developed by the IT department after the preliminary empirical studies. The task for contract approval by the management (COA) was also new to the procedure. The differences in the consignation process are not discussed further in details, because they are not relevant for the actual process model transformation. These differences are mentioned here to show that enterprise work practices and processes are subject to continuous change and need agile, user-centric software support [WR95].

9.3.3.2.3 Findings

A diagrammatic representation of the captured task delegation graph for introduction of consignation sales is shown in Figure 9.3 (the task names are freely translated by the author from German, a screenshot is available in [SS08a]). CSO had received an email from a FE, informing her that a customer has requested consignment sales and asking her to *check the payment history* of this customer. As a result CSO had created a (root) task with several (sub-) tasks for processing the upcoming *consignation application*. Although the *contract preparation* would normally not start prior to a *creditworthiness check*, FE had sent a filled consignation *application form* and *base stock list*, which CSO had attached to her *prepare contract* task before she had *checked the creditworthiness*. CSO commented this exceptional circumstance:

"This is a big customer and we know them for years … Mr. [FE] prepared the documents before having the creditworthiness check to accelerate the process … it [check] was a formality which we needed for completeness later on"

As the *creditworthiness check* was completed (percent change) after the *contract preparation* had started (artifact changes), both tasks were exported as parallel in the jBPM workflow.

CSO had further sent an *approve contract* task to COA, who had completed it. After that CSO had completed the *send contract* task in CTM, which resulted in strict sequence of these tasks in the generated workflow.

CSO had delegated the *archive contract* task to SL1, by asking her to take over and *finalize the processing of the consignation application*. SL1 had delegated a task for the *setup of a folder for automated Excel/IDoc to R/3 conversion* to ITE, who had maintained this task independently from sales by removing unnecessary attachments and increasing the percent complete. SL1 had further requested from SL2 to *enter the customer in the consignation customer list*, and in her to-do list SL1 had created a CTM task for the *creation of a table for regular stock update* with a sub-task for *sending the table to the customer*. Thereby SL1 had attached a *report template*, and later on, a customer-specific *report table* (Microsoft Excel files) in the *create table* task (artifact changes). In the meantime SL2 had marked his task as completed (percent change). In the generated workflow this finally resulted in a fork with three parallel activities as shown in Figure 9.3 (b) (screenshot is available in [SS08a]). SL1 commented the resulting sequence flow:

"It's compelling that it [generated jBPM workflow] shows the tasks in parallel … I didn't think of that previously [during explicit workflow modeling] as that [sequential flow] is the obvious way we do it … we'll certainly perform less efficiently if I don't get the task for preparing the reporting table until Mr. … [SL2] has filled the consignation customers list."

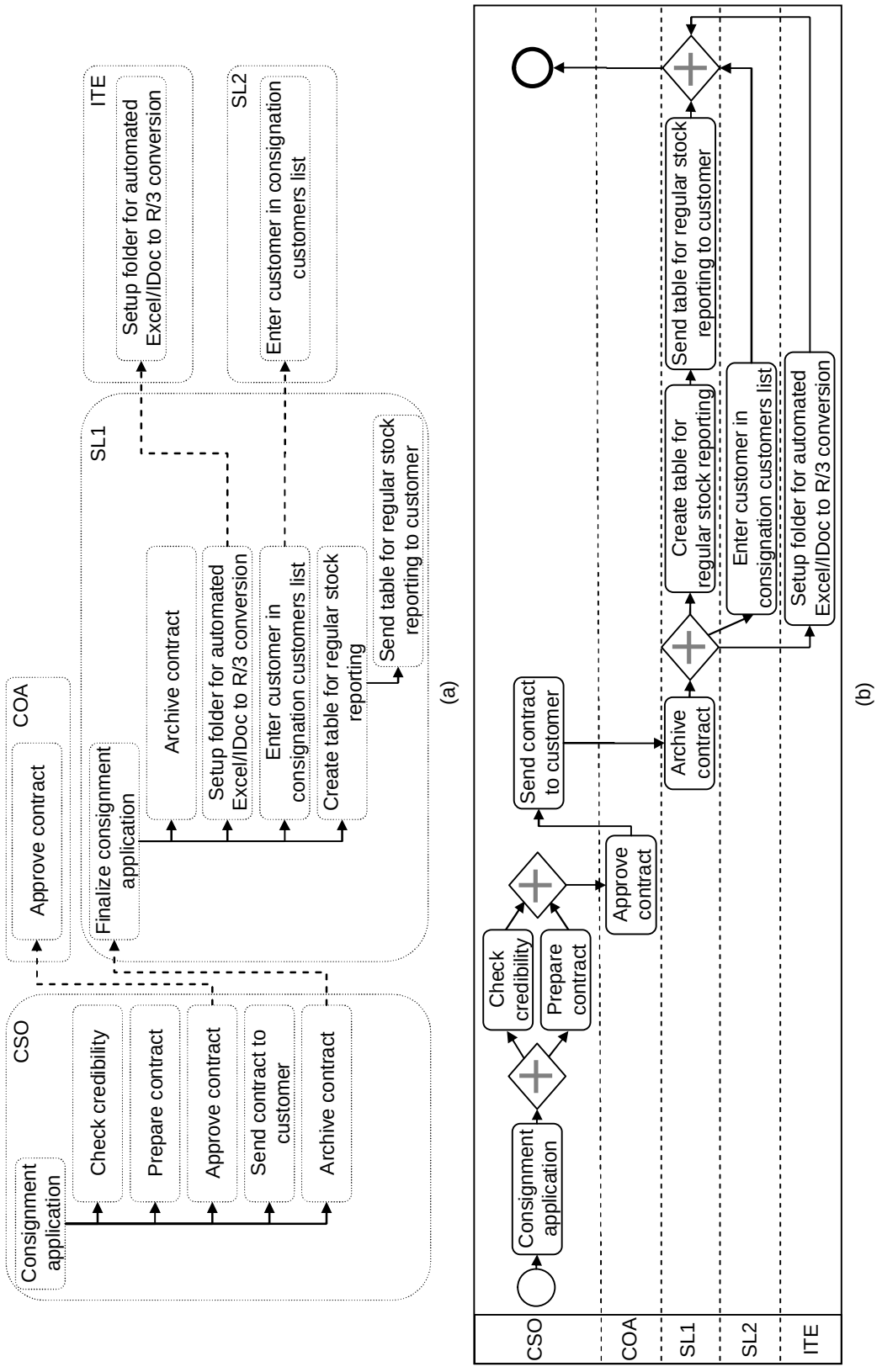


Figure 9.3: Process for initiation of consignation sales: (a) task delegation graph; (b) transformed model

In the generated jBPM workflow the consignation *contract* and *report template* files were set as template - and the other documents as dynamic artifacts. User entries were generated in the user repository during tracking, based on the users' email addresses. Organizational and security roles were assigned to the available users in the user repository by the evaluators through adding the users to the appropriate jBPM user groups. No inconsistent ad-hoc task ranges were detected during the process model transformation.

Later on the generated jBPM workflow was deployed and the users were asked to execute it by maintaining the respective tasks, hence simulating the company-internal process starting with the CSO tasks. All template artifacts were available over artifact links in the workflow tasks' web forms and could be retrieved from the artifacts repository. The case-specific files, like e.g. contract draft, were added through upload of sample files in the web forms of the jBPM task instances. The tasks were assigned correctly according to the user data from the initial ad-hoc process. The provided task sequence did not require deviations.

9.3.3.2.4 Summary of Findings

The case study for process formalization through end-users' involvement shows the adequacy of the provided concepts for transformation of user-defined task delegation graphs to workflow models under increased business collaboration.

Omission of requester tasks (e.g. *approve contract* task of CSO in Figure 9.3 (a)) reduces the task structure for the derived workflow model in an automated and consistent manner. The workflow task nodes resulting from recipients' tasks are correctly assigned to the persons that actually perform these tasks.

Omission of parent tasks that provide only logical grouping (e.g. *finalize consignment application* task of SL1 in Figure 9.3 (a)) reduces the task flow to the tasks that need to be acted upon and represent logical work items.

Transformation of parent tasks to atomic tasks (e.g. *create table for regular stock reporting* of SL1 in Figure 9.3 (a)) resolves inconsistent task decomposition. Namely, during the process model transformation SL1 agreed that the sending of the stock reporting table is a task, which is subsequent to the creation of the table rather than part of the creation task.

Sequence flow detection based on task ranges enables discovery of parallel flow and optimization of the formal process models (see *create table for regular stock reporting*, *enter customer in consignation customers list* and *setup folder for automated Excel/IDoc to R/3 conversion* in Figure 9.3 (b)). Artifact changes and changes to percent complete and status were selected as task processing changes during the transformation.

Document flow and user assignments are transformed in a semi-automatic manner with reduced manual effort.

Although a successful transformation of a user-defined process model to an executable workflow was possible, the end users considered that the involvement of process designers or developers, who can provide guidance and validate the formal models from technical perspective, is intrinsic for the success of such initiatives. In this case study this role was played by the research team. In industrial settings, this role would require engagement of external business process technology experts.

9.3.3.3 Extensions of Workflow Models Through Ad-Hoc Deviations at Runtime

A process for settlement of consignation sales occurred twice in the database backups. As consignment sales reports were sent in the end of each week and consignations were settled each Monday, the process was considered appropriate for automation. In the captured task delegation graphs, the process was defined through a set of high-level tasks. During the debriefing interviews after the long-term test usage, it turned out that the specified tasks do not seem to

reflect this process fully. Hence, a captured task delegation graph of this process was considered suitable for evaluating the refinement of structured workflows through ad-hoc deviation tasks.

9.3.3.3.1 Setting

An exercise for execution and refinement of a process for settlement of consignment sales was performed with SL1, SL2 and CSO who were the stakeholders involved in that process. A workshop was organized with the involved process participants, which started with a 40 minutes tutorial on the jBPM web front-end where explanation was provided to the users how deviations can be handled through creation of ad-hoc CTM tasks. A jBPM workflow was generated from a captured task delegation graph for the consignment sales settlement and deployed on the jBPM engine. Then the users were asked to process a weekly consignment settlement for a customer by maintaining the tasks in the jBPM workflow and deviating where needed. Think-aloud and contextual inquiry [BH98] methods were used to track their strategies and intents. The exercises were videotaped for analysis and partially transcribed later on.

9.3.3.3.2 Process Description

A captured task delegation graph of a process for settlement of consignment sales is shown in Figure 9.4 (a) (task names are freely translated by the author from German, screenshot is available in [SSFM08d]). Two versions of this process were captured in the tracking repository – one with 5 tasks and one with 10 tasks. The discussion in the following is based on the more detailed version. The process is as follows.

SL1 receives a consignment sales report from a customer per email. The report is a CSV (Comma Separated Values) file, describing customer data, such as e.g. International Location Number (ILN), address etc., and consignment sales balance. This report is based on the table for regular stock reporting used by consignation customers (see also Figure 9.2 and Appendix B). SL1 *checks the report for consistency* as wrong input data like ILN can cause errors in the further processing. After that she *enters the sales report data in SAP R/3* system by copying the report in a special folder, from where the file is automatically read into the system. SL1 then describes the *supply for the withdrawn consignment items* in R/3 by specifying type and number of items. Then she asks SL2 to *process the shipment*. SL2 *reserves the amount for shipment* in another transaction in R/3. The data is used by logistics to trigger the re-supply. SL2 further sends a *feedback about the completeness* of the settlement to CSO for accounting purposes. CSO receives the feedback and later on *checks the payment* for the re-supplied goods.

The captured task delegation graph does not provide detailed description of all relevant cases that were detected during the preliminary empirical studies and that are represented in the Hierarchical Task Analysis (HTA) diagrams (cf. Appendix B). The captured task delegation graph refers to the processing of a concrete business case and reflects only the tasks and collaborative activities in a single ad-hoc process instance. Logical differences between the tracked process and the procedure discussed during the HTA analysis can be found. For example, the tasks related to explicitly documenting the customer transactions in a transactions tracking list (cf. Figure B-5) were not available in the task delegation graph. On the other hand, the task delegation graph contained tasks about completeness feedback to accounting and payment checking which have not been stated during the HTA analysis. This again points at the increased variability enterprise processes and at the need for agile, user-centric process support [WR95].

9.3.3.3.3 Findings

The jBPM workflow which was derived from the captured task delegation graph contained the tasks in a strictly sequential order as shown in Figure 9.4 (b).

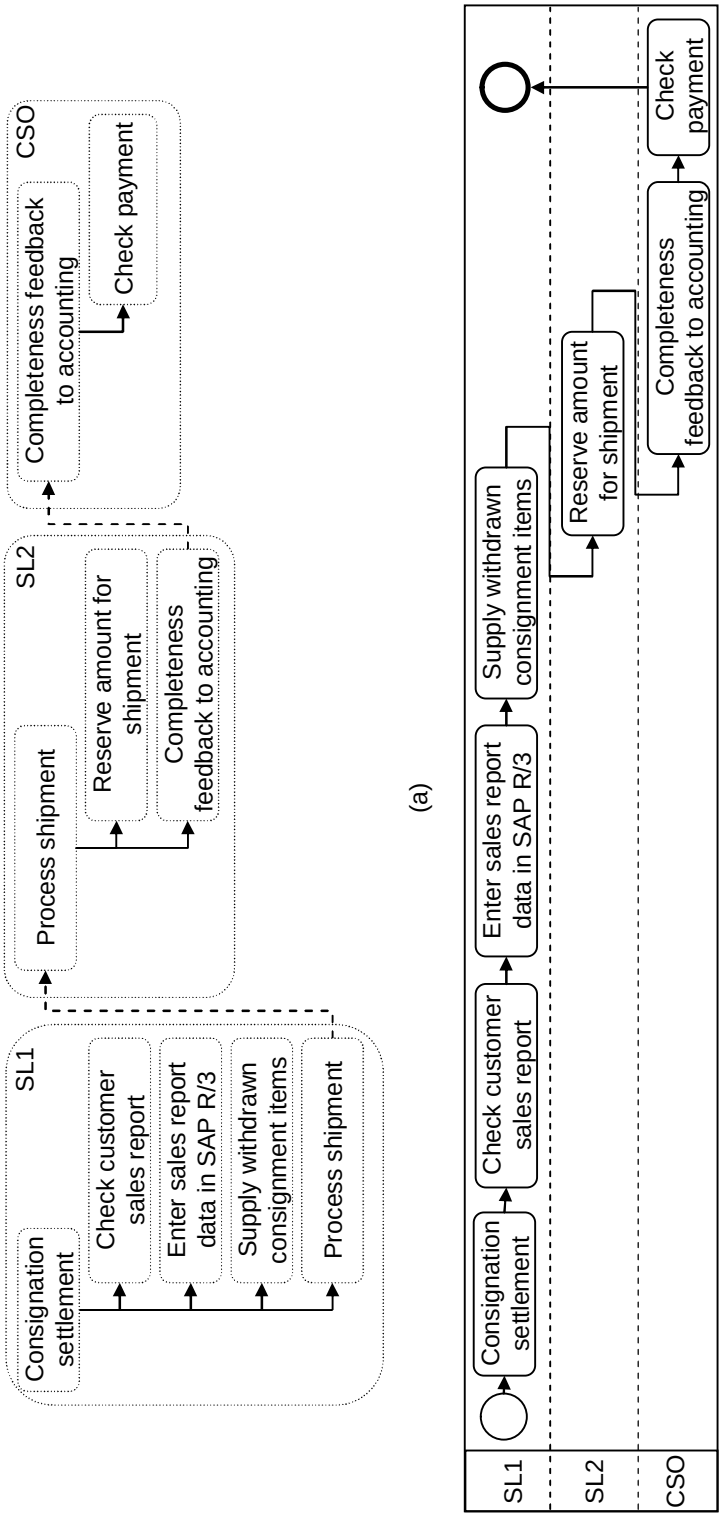


Figure 9.4: Process for settlement of consignation sales: (a) task delegation graph; (b) transformed model

Artifact changes and changes to percent complete and status were selected as task processing changes for the transformation. No inconsistent ranges were detected during the transformation. SL1, SL2 and CSO were asked to process a weekly consignation settlement for a customer by maintaining the corresponding tasks in the jBPM workflow and deviating where necessary.

After SL1 transferred the data from the customer sales report to the R/3 system, she cross-checked the resulting invoiced amount in the system with the amount in the sales report. There was a slight difference in both sums. SL1 commented:

"Yes, sometimes the reported customer prices differ from our company prices … this is mostly due to the different calculation of taxes as customer calculates per delivery and we per item … Well, as in this case it is usually a matter of cents … we continue the settlement with the customer prices and ask Mrs. … [COA] to contact the customer and request them to correct the prices for the next settlement."

The differences were minimal and were considered insignificant. As a result SL1 deviated from the currently started jBPM task *enter sales report data in SAP R/3*, and created a CTM task in her to-do list with the same name. She then created a sub-task *cross-check invoiced amount* and to this subtask she added another subtask *ask customer for correction*, which she delegated to COA. This practice differed from the procedure described during the HTA analysis, where sales support would clarify the differences with the customer (cf. Figure B-5). Currently, COA served as a single point of contact to corporate customers (clarifications with small customers were still done by sales support). Observed deviations from the derived workflow are shown in Figure 9.5 (a). As the process could in this case continue (with customer prices), SL1 returned back to the deviated jBPM task and completed it. She then completed the *supply withdrawn consignment items* task without deviations.

When SL2 started the *reserve amount for shipment* task he inspected the data about previous deliveries in R/3 and the reported amount of sold items in the customer sales report. For one of the consignment items he noticed that the reported sales exceeded the previously delivered amount. SL2 explained:

"We ship this item per store and I assume that the customer has transferred items between their stores, without notifying us. … I'll need to inform Mrs. …[CSO] so that she can issue liability statements for the excess."

SL2 considered that such inconsistencies would be propagated with the *completeness feedback* to CSO, so he entered a comment in the jBPM workflow, explaining the inconsistency. A further consignment item needed to be shipped as a set of multiple, smaller items. In the concrete case, items from the set were not delivered to the customer in the required amount and had to be re-supplied additionally. SL2 commented:

"Sets are often requested with different content from different customers … we have to adapt and deliver the set items on demand."

SL2 hence deviated from the started *reserve amount for shipment* task in the workflow and created an ad-hoc task *order set items* in his to-do list, which he set to status in progress. SL2 commented that the procedure for shipping the set items is the same as for all other items, and that shipment of set items is handled independently, as a special case in the overall consignation re-supply. Thus, SL2 reserved the shipment of the currently handled consignment items in the R/3 system, and returned to the deviated jBPM task to complete it, so that CSO can handle further the consignation settlement. SL2 then started processing the order of the set items.

When handling the *completeness feedback* task in the jBPM workflow, CSO read the comment

of SL2 about the inconsistency in delivered and sold consignment items. CSO commented:

"I need to create liability statements for that [inconsistency] so that the customer can correct the problem on their side."

CSO created an ad-hoc task *prepare liability statement* in the to-do list and started preparing the document. When she was ready later on, she returned to the jBPM workflow to check again the comment for the *completeness feedback* task and to see if all mentioned issues have been reflected through appropriate liability statements. Then CSO completed the active *completeness feedback* task before completing the *prepare liability statement* task in her to-do list.

For the missing set items, CSO later on received a delegated CTM task *completeness feedback* from SL2, who had reserved the shipment for these items. CSO accepted the ad-hoc *completeness feedback* task and created again a *check payment* sub-task for the set items. It was not possible to follow the processing of the *check payment* tasks of CSO in the jBPM workflow and in the ad-hoc task delegation graph from the *order set items* deviation as these tasks required customer actions. But CSO agreed that this would end the consignation settlement process and completed the tasks.

Finally, the jBPM process model was regenerated with all available data from the initial task delegation graph and from the execution of the jBPM workflow with deviating tasks, i.e. under the supervision of SL1, SL2 and CSO, with who the export modes of ad-hoc tasks were discussed. No inconsistent task ranges occurred during the regeneration. The *order set items* task was exported as parallel sub-process whereas the other deviations were exported as sequential workflow tasks. The parent task *completeness feedback to accounting* from the initial task delegation graph was exported as atomic task preceding the sub-tasks sequence. As result the *check payment* task preceded the *prepare liability statements* task form the deviation. As the *check payment* task was considered as the final task in the process, it was manually moved after the *prepare liability statements* task by shifting the corresponding edges. The resulting process model is shown in Figure 9.5 (b) (screenshot is available in [SSFM08d]) and the sub-process for the order of the set items is shown in Figure 9.5 (c). Users appreciated having the complete workflow with all possible deviations in it. CSO commented:

"If the reported balance is ok, I'll just complete this task [liability statement] straight away … but I certainly want to have it there to make sure I won't forget it."

Users considered that the jBPM workflow functionality can bring benefits to them as the automated task assignment would save them the effort to distribute tasks per email as usual. They further reported that they consider the final workflow real-life compliant and would try to use it on regular basis and possibly to develop several variations for different customers.

9.3.3.3.4 Optional Deviations – a Concept Extension for Optimized Deviation Flow

A critical view on the derived workflow model reveals that it can be significantly improved through handling of exceptional cases through alternative flows. Branches can be considered for the *ask customer for correction* and *prepare liability statements* tasks, and for the *order set items* sub-process, as these tasks and sub-process may not be necessary in the overall flow if the customer data is consistent and no set-items need to be ordered additionally.

Optional deviation flow is introduced as conceptual extension for the optimization of workflow models that are refined through deviations. This conceptual extension basically means that during process model transformation the *transformation control* should enable users to select if the deviation tasks are optional, i.e. if the deviation flow should be added in alternative flow branch. Thus, through such additional export mode option users are enabled to produce optimized workflows with decision branching points for optional deviation flows.

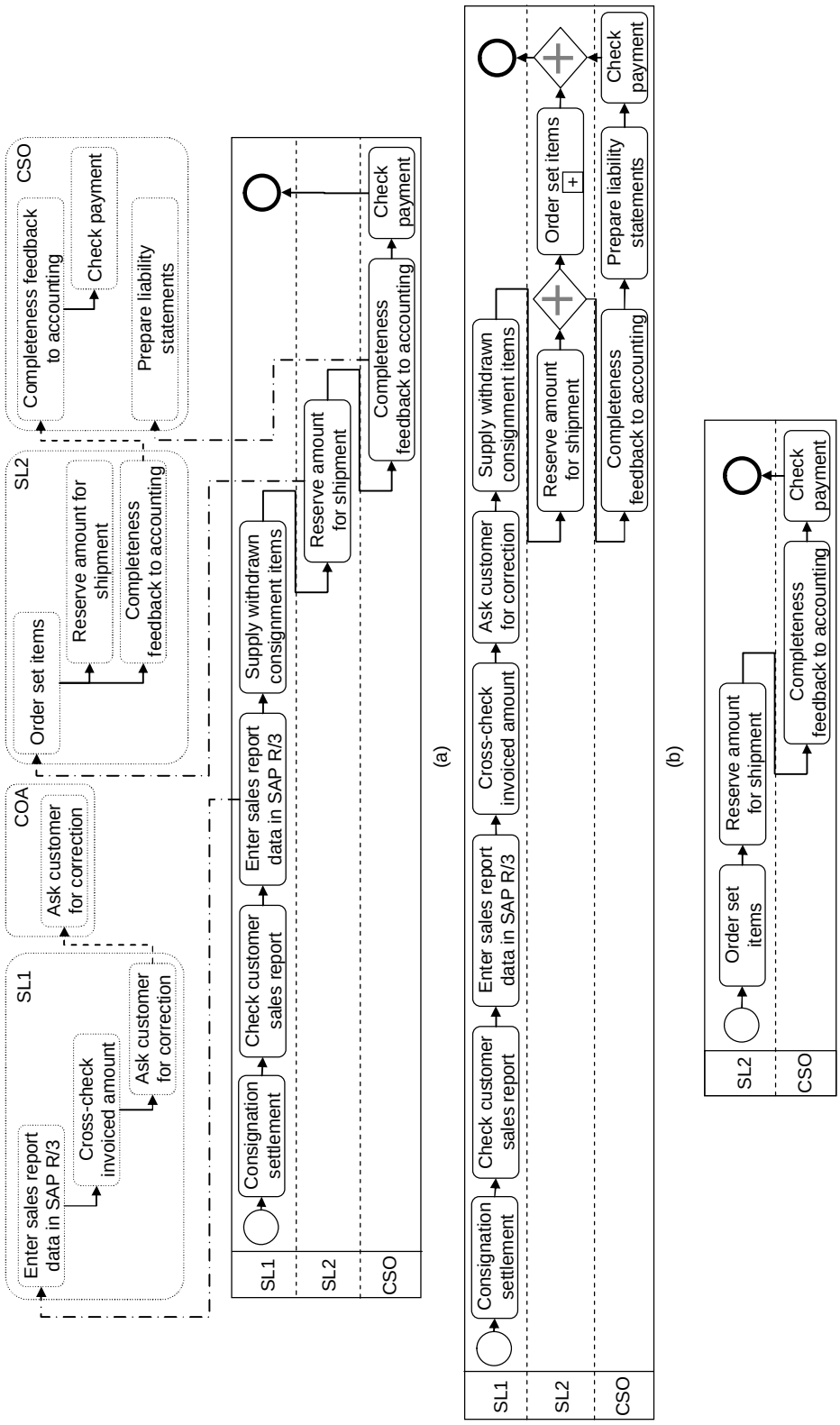


Figure 9.5: Process for settlement of consignation sales: (a) workflow instance with deviations (chain-dotted arrows); (b) redefined workflow; (c) transformed sub-process

9.3.3.3.5 Summary of Findings

The evaluation of the extension of formal workflow models with ad-hoc task deviations at runtime showed that the chosen approach enables end-user driven refinement of underspecified workflow models. The complementation of the formal workflows is accomplished in a semi-automated manner with minimal manual effort by choosing export mode options of ad-hoc tasks.

To optimize the derived workflow models with deviation flow, the thesis introduces the conceptual extension of *optional deviation flow*. This conceptual extension means that during process model transformation the transformation control functionality enables users to specify if the extension flow should be added as an optional flow. In complex cases, the refinement of workflow models may require assistance from business technology experts such as process designers or developers, who can assist during the transformation and validate the formal model. These experts can work collaboratively with end users to develop consistent process models by referring to user-defined workflow extensions resulting from deviations and fitting the deviation flow in the initial workflow model.

9.3.3.4 Summary of Findings - Composition and Refinement of Structured Workflows

The evaluation of the composition and refinement of structured workflows has shown that end-user tailors can successfully transform weakly-structured task delegation graphs to formal workflow models, by using complementary representations of user-defined, ad-hoc tasks and formal workflow models. The transformation of ad-hoc processes to formal workflows benefits from multiple representations and fosters tailoring as collaboration between users with different domain expertise and technical skills. Thereby technically-inexperienced business users are involved implicitly in process modeling by using their user-defined task hierarchies during the workflow model derivation. Selection of export mode options enables flexible interpretation of user-defined task delegation graphs and generation of formal workflow models with decreased manual effort.

Deviations from formal workflows during execution are enabled with on-demand, ad-hoc task hierarchies. These deviations successfully enable end-user driven process model refinement. To support the optimization of refined workflow models with alternative flows the conceptual extension of optional deviation flow has been proposed.

The level of specificity of a structured workflow model depends on the level of specificity of user-defined weakly-structured process models. While business users can deliver the basic diagram of a business process model, the assistance of process designers and developers can be needed to validate the user-defined models from technical perspective. Composition and refinement of structured workflows thus relies on the effective collaboration between business users and business technology experts.

9.4 Evaluation Based on the Technology Acceptance Model

The final evaluation phase provides assessment of the developed conceptual framework based on the Technology Acceptance Model (TAM) [Dav85, Dav89]. This phase has the purpose to perform a combined, quantitative evaluation of the elaborated concepts by referring to various components of the CTM system. The evaluation provides a combined assessment about the acceptance of end-user driven business process composition from end user's perspective.

9.4.1 Setting

The TAM-based evaluation was performed in two of the industrial partner companies – TXTL and ASPL. Details about the involved users and their experience with CTM are provided in the next sections.

9.4.1.1 Users in TXTL

After the long-term evaluation phase, the research project time-frame allowed further usage of the CTM system in TXTL. The user group in this company was extended through a Sales Management Assistant (SLA). She received a 40 minutes individual introduction to CTM, followed by a 30 minutes training in the basic functionalities. The introduction and tutorial involved also the jBPM workflow export functionality.

SLA used the tool for about two weeks along with the other participants before the TAM-based evaluation took place. During this period SLA had created two CTM tasks with sub-tasks and inspected a task delegation graph and a dialog. Through this 7 users in TXTL had hands-on experience with CTM. The users had used CTM for different periods. These differences were considered during the evaluation.

9.4.1.2 Users in ASPL

Additionally, 6 test users for CTM were recruited in ASPL. The users were selected for having related collaborative tasks and no formal IT education. Particularly, the 6 participants were involved in the transfer of mature prototypes from prototyping to serial production, which was also the process, envisioned for optimization through CTM during the preliminary empirical studies (cf. Chapter 2). The user roles are briefly described in the following:

- *Team Leader in Product Management (TLPM):* coordinates the development of new and existing products, coordinates order management for products.
- *Production Readiness Employee (PRE):* elaborates preliminary and final costing, creates item lists and work plans, records wastes, performs time recording in production.
- *Clerk in Order Management (COM):* participates in disposition, production planning, personnel planning.
- *Purchase Employee (PE):* performs order management, checks prices and suppliers on regular basis, applies new raw material numbers.
- *Customer Service Employee (CSE):* manages customer orders, coordinates order management with the other departments (quality assurance, product management).
- *Quality Management Employee (QME):* performs product validation, handles reclamations (from customers and to suppliers).

9.4.1.3 Test Usage in ASPL

Due to the limited time-frame of the research project, only a 3 weeks test-usage of CTM was possible in ASPL. The trial was initiated with a workshop in which a 90 minutes presentation on CTM was given, followed by 40 minutes individual tutorial of each user on the basic CTM functionalities. The introduction and tutorials included the jBPM workflow export functionality. After several days the users were visited individually, to check how they are working with the tool and to provide further instructions.

9.4.1.4 Questionnaire

A TAM-based questionnaire focusing on *perceived usefulness* and *perceived ease of use* for the major CTM components was completed by 13 CTM test users (7 from TXTL and 6 from ASPL). The assessed CTM components are discussed in details later on. The questionnaire contained the standard TAM questions [Dav85, Dav89] on usefulness and ease of use in two different tables for

each assessed CTM component – one table for usefulness and one for ease of use. The questionnaire asked participants to rate the extent to which they agree with each statement by putting an "x" mark in a column from -3 "Strongly Disagree" to + 3 "Strongly Agree" with 0 as the "Neutral" answer.

Perceived usefulness was considered as a measure that can be obtained for all assessed CTM components and underlying concepts as all 13 users were exposed to all assessed components, i.e. even users who did not use the system extensively were educated in its major functionalities during the tutorials and individual exercises. The questionnaire explicitly suggested that the participant should open the respective CTM component and try a provided set of basic operations as a reminder of the generic purpose and functionality of this component, before filling out the respective questions for the component (both for usefulness and for ease of use). Alternatively, the participants were instructed that they can skip usefulness questions for a given component if they feel that they do not understand its purpose or the generic functionality that it provides. Thus, the users were expected to be sufficiently familiar with the system components when answering the questions related to usefulness. According to [Dav85] usefulness does not require extensive hands-on experience and can be estimated based on *observation* of a provided functionality, which can be even video-based.

Ease of use was evaluated only for those components, with which the users had actually worked. The users were asked to skip the ease of use questions for those components that they have not used individually at least once after the tutorials and exercises.

Self-efficacy, **benefits** and **drawbacks** were assessed through additional questions at the end of the survey. The purpose was to provide a concluding assessment of the individuals' perception of whether they can successfully perform and participate in process composition and of their judgment about possible benefits and drawbacks from such process composition.

Background information of the participants was enquired in the final section of the questionnaire. The participants were asked to provide details such as current job qualification, position in the company, number of people they manage, education, and working years in the company. These details were used later on to assess the participants' profiles.

9.4.1.5 Interviews

Immediately after filling the questionnaire, each participant was interviewed. The interviews were semi-structured and followed the sections of the questionnaire. By going through the different CTM components the participants were asked to what extent the provided functionality and the generic concepts behind that functionality could be useful and could help them in their daily work. The interviews helped to enrich the quantitative assessments with qualitative data. This data revealed different user types and varying roles and needs affecting the end user's attitude towards business process composition.

9.4.2 Findings

The introduced holistic concept of end-user driven business process composition encompasses several major aspects which have been discussed throughout the dissertation:

- *Personal task management in a light weight to-do list* (R1) is the primary mean for involving end users in business process composition.
- *Exchange of tasks and deliverables over email* (R2) binds the individual task hierarchies to overall task delegation graphs by additionally capturing task-related email exchange in task delegation dialogs for quick, aggregated overview of task-related email messages.
- *Task delegation graphs* and *task delegation dialogs* provide enhanced transparency into evolving collaborative tasks (R3) and structured storage of process information.
- *Task patterns* enable exchange, adaptation and reuse of process knowledge (R4).

- *Task evolution* tracing is enabled through ancestor/descendant relationships between task instances and patterns, and enables structured comparison between running processes and best-practices (R5) and between different best-practice variations (R6).
- *Process transformation* of task delegation graphs to structured workflow models enables process formalization and automation (R7).

The discussion in the following focuses on the various concepts for end-user driven business process composition through the respective CTM components. Considering the small sample size (N=13), a Shapiro-Wilk normality test [SW65] was performed for the overall usefulness scores for each component. The test results for all components ($p < 0.05$) indicated a non-normal distribution of the sample scores. Hence, nonparametric statistics are used of the result analysis.

9.4.2.1 Concept Assessments Based on TAM

The measurement level of rating scales (ordinal or interval) is a subject of ongoing debate in research literature [BD06]. Interval level of measurement for rating scales is commonly assumed to allow inferential statistics [BD06]. Related research reports evaluations based on the TAM where usefulness and ease of use ratings are treated as interval measures [BBPS06, RC07]. Interval level of measurement is assumed also in the dissertation.

The questionnaire responses were entered into Microsoft Excel and then loaded in SPSS (originally Statistical Package for the Social Sciences) [SPSS08] for further statistical analysis. Descriptive statistics are provided in Appendix D with mean (M) and standard deviation (SD) values, and the number of valid responses (N).

According to [Dav85, Dav89] the usefulness and ease of use questions from the TAM evaluation questionnaire form distinct clusters which refer to various usefulness and ease of use aspects. The aspects related to usefulness are shown in Table 9.2.

Table 9.2: TAM question clusters and related usefulness aspects [Dav85, Dav89]

	Cluster	Aspect
Usefulness	A	job effectiveness
	B	productivity and time savings
	C	importance of the system to the users' job
	D	control over the job

This section provides a detailed analysis of the various usefulness aspects for each of the concepts for end-user driven business process composition based on the related CTM components. Ease of use is assessed only for components that were actually used by end users to evaluate the light-weight character of the provided implementation. Light-weight support is considered important with respect to ensuring a gentle slope of complexity [MCLM90] for process tailoring.

9.4.2.1.1 Personal Task Management

The concept of involving end users in business process composition based on personal task management in light-weight to-do lists (R1) was evaluated based on the user acceptance of the CTM to-do list (cf. Figure 8.1).

Usefulness

An overall usefulness mean score of 1.08 resulted for the CTM to-do list (all results are based on a -3 to +3 scale, cf. Table D-1). A dichotomous variable (0 or 1) was used to indicate whether a study participant has been using the Microsoft Outlook to-do list before CTM was introduced.

This dichotomous variable was used as a grouping variable to perform a Mann Whitney U test [MW47] (in the following referred to as MW test for brevity) on the overall usefulness scores. The test result ($U=17$, $N_1=5$, $N_2=8$, $p_{(exact)}=0.724$, two-tailed) showed insufficient evidence to conclude that there is a different usefulness perception for the CTM to-do list between persons that had been using Microsoft Outlook tasks before the CTM installation and those that were not familiar with the Outlook to-do list functionality.

The interviews later on revealed that some of the participants (especially in ASPL) were simply unaware of the standard Microsoft Outlook to-do list functionality before the CTM installation. In TXTL on the other hand Outlook tasks were extensively used and even delegated by senior employees before the CTM installation. Once all users were aware of the provided to-do list functionality, similar usefulness estimations appeared for both user groups.

A further MW test was performed to check the dependence of the usefulness estimation for the CTM to-do lists between participants which were using the CTM system for a longer period of time (all 6 participants from the preliminary and long-term evaluation in TXTL), and those who were exposed to it for a short period (SLA from TXTL and all 6 users from ASPL). The result ($U=19.5$, $N_1=6$, $N_2=7$, $p_{(exact)}=0.836$, two-tailed, cf. Table D-11) shows that there is insufficient evidence to assume that persons that were exposed longer to the CTM system have different usefulness expectations towards the CTM to-do list than those that have used the CTM system for a shorter period of time.

Spearman correlation tests were further performed to assess whether there is a dependency between the usefulness estimation for the to-do list and the number of persons managed by a given study participant, or between the usefulness estimations for the to-do list and those estimations for other components. The latter aspect was tested to check if given concepts and components address similar user needs. The results are summarized in Table D-13. The results did not identify any significant correlations for the to-do list usefulness estimations.

Comparative Usefulness Cluster Overview

A comparative overview of the usefulness estimations for the different question clusters (cf. Table 9.2) is shown in Figure 9.6. The questions from clusters A, B, and D have positive means > 0.5 on a -3 to +3 scale. Thus to-do list applications are considered useful with respect to the corresponding aspects from Table 9.2.

Job productivity and time savings (cluster B) is an aspect where very few or no usefulness is expected from the to-do list functionality. The low usefulness expectation is indicated through low positive mean scores of 0.23 for questions 3 and 5 and a negative mean score of -0.38 for question 7. During the interviews, it became clear that the users do not consider task management applications as working applications that deliver results or complete users' job, but rather as assisting applications which help to organize the job. A typical use case for the to-do list was described by PRE:

"If I need to make a monthly report then it will be nice to get a reminder, for example in which month in which day I should do the report. Then it [to-do list] will be very helpful."

In the discussed case however, the report would still be done in the corresponding working applications (Microsoft Excel and Word). Thus the low usefulness estimation can be explained through the fact that users do not immediately relate task management with productivity.

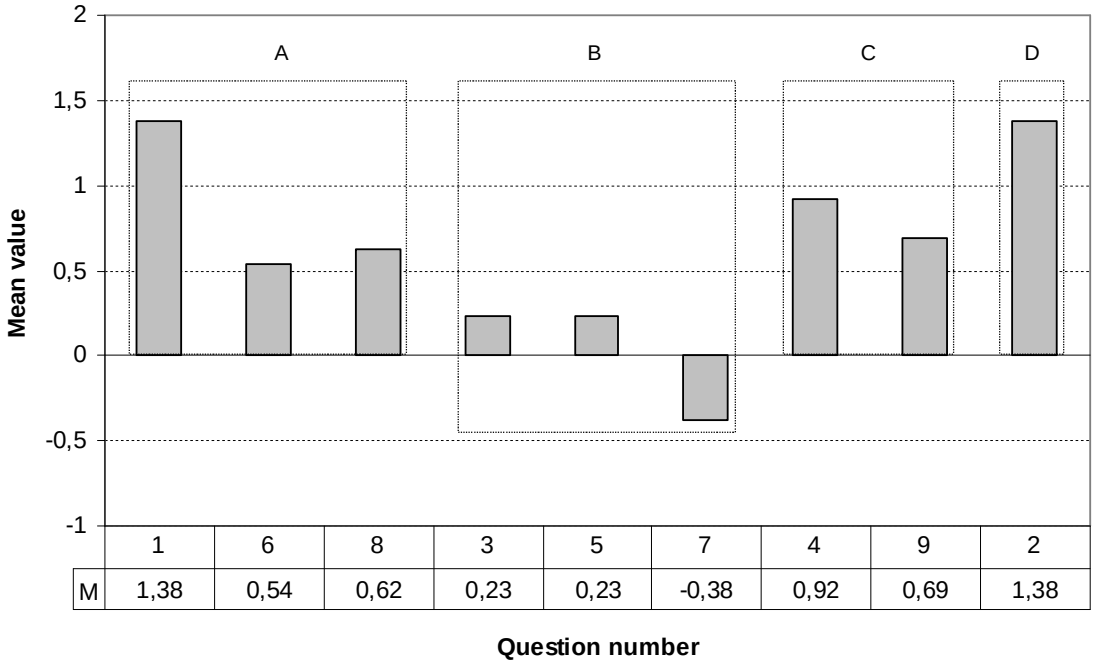


Figure 9.6: Comparative overview of usefulness estimation for **to-do list** according to question clusters - question numbers with mean value for each question are given on the x-axis

Ease of Use

Ease of use estimation results for the CTM to-do lists are provided in Table D-2. All participants in the evaluation used the to-do list. Thus the results are obtained from 13 responses for each question. The results show positive ease of use perception, where negatively formulated questions are met with disagreement and positively formulated questions have received positive answers.

A MW test was performed to check the dependence of the ease of use estimation of the CTM to-do list between participants which were using the Microsoft Outlook to-do list prior to the CTM installation and those who have not. The results ($U=10$, $N_1=5$, $N_2=8$, $p_{(exact)}=0.171$, two-tailed) show insufficient evidence to conclude that the ease of use estimations for the CTM to-do list are different between both user groups. This observation confirms the light-weight character of the to-do list functionality and the gentle learning slope. Further support for the light-weight character was provided through the fact that no significant differences between the overall ease of use estimation for the CTM to-do list of persons exposed to long-term ($N_1=6$) and of persons exposed to short-term ($N_2=7$) CTM usage could be found (cf. Table D-12).

9.4.2.1.2 Process Overview

The concept of providing enhanced transparency in evolving collaborative processes beyond the personal workspace (R3) through task delegation graphs was evaluated through questions on the task delegation graph overview (cf. Figure 8.3).

Usefulness

The task delegation graph overview received a positive overall usefulness mean score of 1.15 (cf. Table D-3). A MW test was performed to test the difference between the usefulness estimations of persons that have been using CTM during the long-term evaluation and those that were exposed only to a short-term CTM usage (cf. Table D-11). The result is significant at the 0.05 level ($U=5$, $N_1=6$, $N_2=7$, $p_{(exact)}=0.022$, two-tailed) and reveals differences between both user

groups. A closer analysis of the results revealed that users, exposed to long-term prototype usage gave lower usefulness scores (M=0.5, SD = 1.22) than the users exposed to short-term usage (M= 1.71, SD = 0.49). While these answers could be explained with novelty effects, the results for the other components did not provided support for such assumption. Namely, novelty effects would affect all components which were not currently provided in the users' working applications such as delegation dialogs or task patterns.

A possible explanation for the lower usefulness scores after long-term usage was detected later on during the interview analysis. In particular, all users that were exposed to the short-term usage except SLA were from the ASPL company. The limited project time-frame did not allow for capturing actual processes for transfer of ready prototypes from prototyping to production. This process was targeted as a central use case for CTM usage in ASPL. Thus, the users had expectations that the task delegation graph overview would be substantially useful in the targeted process. On the other hand, several processes could be captured in TXTL. Concerns regarding increased usefulness have emerged after the actual usage. These concerns were related to the currently missing authentication for inspecting task delegation graphs and to potential pitfalls in the provided transparency and decision support if users do not manage task status and information on regular basis. These aspects were considered especially critical if the system had to be used from a broader user audience.

A Spearman correlation test between the number of managed persons and the overall usefulness estimation for task delegation graphs did not reveal any significant correlations (cf. Table D-13). This extends the observations from the preliminary evaluation phase, where primarily employees with managerial positions reported that they want to have an overview. Table D-13 further shows that there are no correlations between the usefulness estimations for the task delegation graph overview and for other components. Thus no overlapping aspects of task and process management are addressed through the task delegation graph overview and the rest of the components, i.e. specifically the task delegation dialog overview. Therewith, a separate user attitude towards transparency in control (task) flow and collaboration flow becomes evident.

Comparative Usefulness Cluster Overview

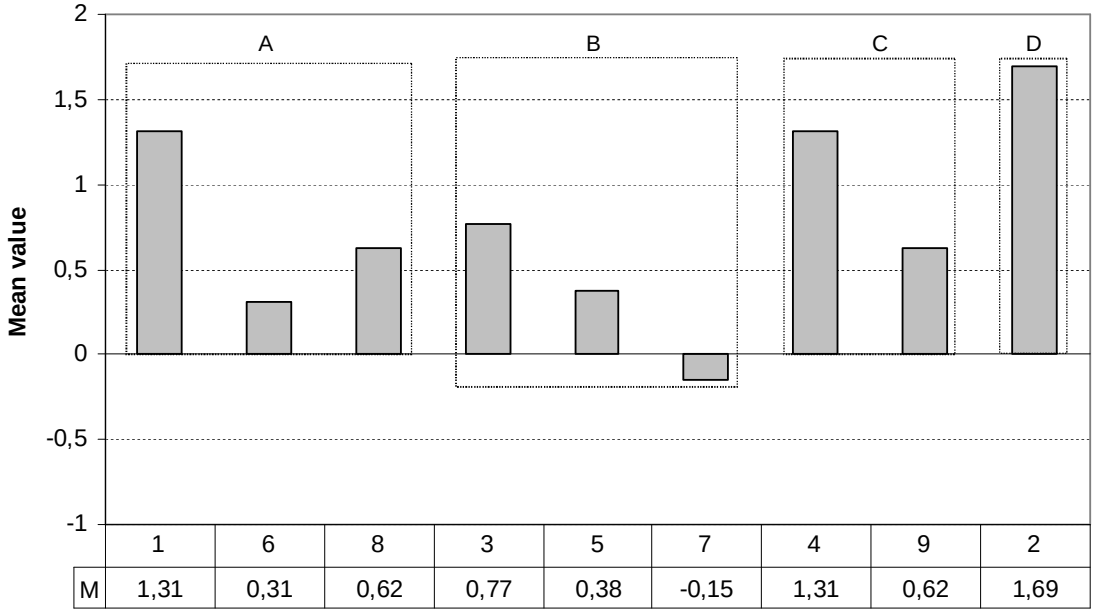
A comparative overview of the usefulness estimations for the task delegation graph overview according to the different usefulness question clusters (cf. Table 9.2) is shown in Figure 9.7. The questions from clusters A, B, and D have positive means, where the *control* aspect (cluster D) has the highest usefulness score.

Positive mean scores for questions 3 and 5 indicate that *job productivity and time savings* (cluster B) is also an aspect where usefulness from the task delegation graph overview is expected. The negative mean value for question 7 however, points that by having a task delegation graph overview, users do not expect to do more work than otherwise possible. This could be again explained through the overall attitude of the study participants, that task management and process management applications do not directly deliver results for the users' job, but rather help to organize and plan the activities, i.e. the task delegation graph overview was considered an assisting rather than as a working tool. This observation agrees with the high score on question 2 related to *control* (cluster D).

Ease of Use

The ease of use estimation results for the task delegation graph are provided in Table D-4. All participants in the evaluation used the task delegation graph overview, thus the results are obtained from 13 responses for each question. The results show positive ease of use perception. A MW test was performed to check the dependence of the ease of use estimation of the task delegation graph overview between participants which were exposed respectively to long-term and short-term CTM usage. The results ($U = 15.5$, $N_1 = 6$, $N_2 = 7$, $p_{(exact)} = 0.45$, two-tailed) show

insufficient evidence to conclude, that the ease of use estimations for the task delegation graph overview are different in both user groups. This confirms the light-weight character of the task delegation graph overview.



	1	6	8	3	5	7	4	9	2
M	1,31	0,31	0,62	0,77	0,38	-0,15	1,31	0,62	1,69

Question number

Figure 9.7: Comparative overview of usefulness estimation for **task delegation graphs** according to question clusters - question numbers with mean value for each question are given on the x-axis

9.4.2.1.3 Dialog Overview

The concept of task delegation dialogs for supporting email-based exchange of tasks and deliverables (R2) and enhanced transparency in collaborative processes (R3) was evaluated through questions on the task delegation dialog overview (cf. Figure 8.2).

Usefulness

The dialog overview received a positive overall usefulness mean score of 1.31 (cf. Table D-5). The inferential tests did not reveal any differences between the usefulness estimations for the dialog overview of users that were exposed to long-term CTM usage and of those that used CTM only for a short period of time (cf. Table D-11). Furthermore, no correlations between the number of managed persons and the usefulness estimations for the dialog overview were detected (cf. Table D-13). Thus, the dialog overview addresses equally the needs of users with various areas of expertise and positions in the enterprises. No correlations were further found between the usefulness estimations for the dialog overview and for other components (cf. Table D-13).

Comparative Usefulness Cluster Overview

A comparative overview of the usefulness estimations for task delegation dialogs for the different question clusters of the TAM model (cf. Table 9.2) is shown in Figure 9.8.

Positive means can be seen for the questions from all clusters. The positive results for the *productivity* cluster (B) could be explained through the fact that all users were involved in collaborative processes, and dealt with increased number of emails. Aggregating relevant emails for a given task to a single dialog instance could reduce email search and sorting efforts. *Control*

over the job (cluster D) reveals as the aspect with highest usefulness expectation regarding dialog overview. This again implies that the provided task management and overview functionalities are seen primarily as means to control and manage work.

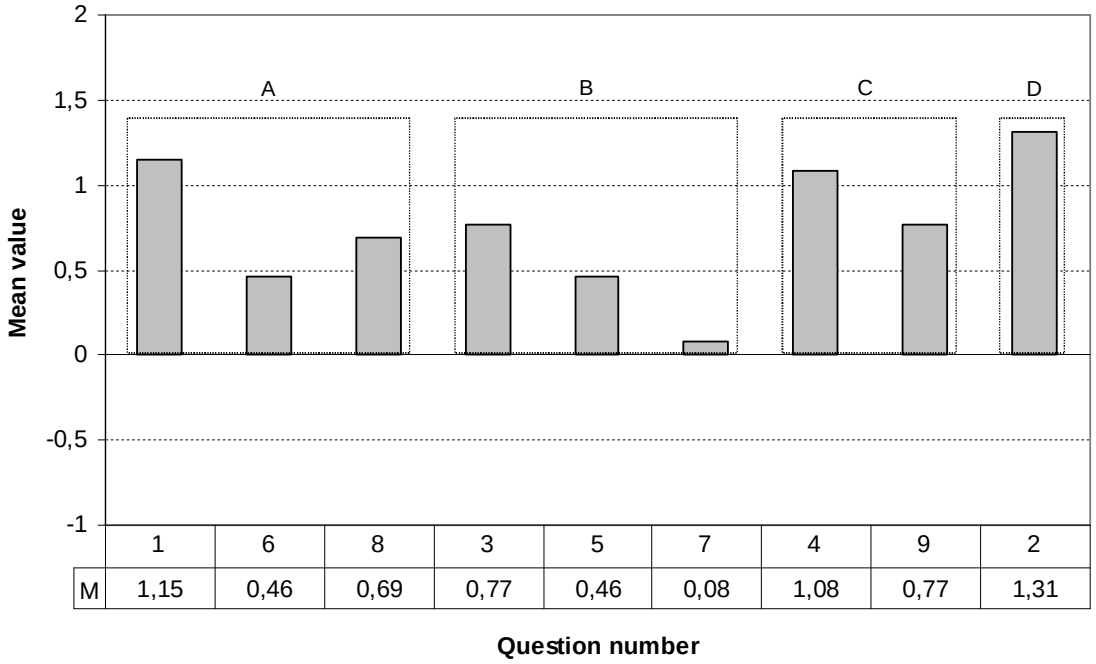


Figure 9.8: Comparative overview of usefulness estimation for **task dialogs** according to question clusters - question numbers with mean value for each question are given on the x-axis

Ease of Use

Ease of use estimation results for task delegation dialogs are provided in Table D-6. All participants in the evaluation had used the dialog overview. The results are based on 13 responses for each question. The results show positive ease of use perception with negative and positive scores respectively for negatively and positively formulated questions.

The light-weight character and gentle learning slope of task delegation dialogs is confirmed through results from a MW test ($U=20$, $N_1=6$, $N_2=7$, $p_{(exact)}=0.95$, two-tailed) showing that there are no significant differences between ease of use estimations of users that were engaged respectively in long-term and short-term CTM usage.

9.4.2.1.4 Task Patterns

The concept of task patterns for enabling exchange, adaptation and reuse of process knowledge (R4) was evaluated through assessment of different aspects of the task pattern functionality.

Usefulness

Usefulness assessments were enquired by referring to the overall task pattern functionality including extraction and application of task patterns from the to-do list. Usefulness estimation results are provided in Table D-5. Although not all users had engaged with extraction or definition of task patterns, these functionalities were introduced to the test users in details during the CTM tutorials and exercises. 13 valid responses for the usefulness assessment were received.

The task patterns functionality received a positive overall usefulness mean score of 0.77. A MW tests did not show any difference in the usefulness assessment of users which were exposed respectively to long-term and short-term CTM usage (cf. Table D-11).

A Spearman correlation test was performed to check for correlations between the number of managed persons and the overall usefulness estimations for task patterns (cf. Table D-13). A negative correlation between both variables was discovered ($r_s = -0.594$, $p = 0.32$, 2-tailed). This means that increasing number of managed persons decreases the usefulness estimations for task patterns. The result from the correlation tests agrees with the findings from the preliminary evaluation, where managerial employees expressed skepticism regarding task patterns. CSO for example was uncertain about the scalability of task patterns in long-term perspective (cf. Section 9.2.2.6). ITL on the other hand reported that he cannot benefit much from task patterns himself, as he is engaged in tactical tasks where he does not see much reuse potential or possibility to benefit from someone else's knowledge in the enterprise (cf. Section 9.2.2.5). Similar concerns were expressed also by managerial employees in ASPL. The conclusion can be drawn that task patterns are considered more useful by employees without managerial functions who can use them as guidelines to ensure that they are performing their (operational) tasks according to the common, recommended procedure.

Comparative Usefulness Cluster Overview

A comparative overview of the usefulness estimations for task patterns according to the different TAM question clusters (cf. Table 9.2) is shown in Figure 9.9. Positive mean scores were received for the questions from all clusters. Thus task patterns are considered useful for all relevant aspects. Of particular notice in Figure 9.9 are the high positive mean scores for usefulness estimations for the *productivity* cluster B. Thus reuse is seen as a direct mean to increase performance and realize time savings.

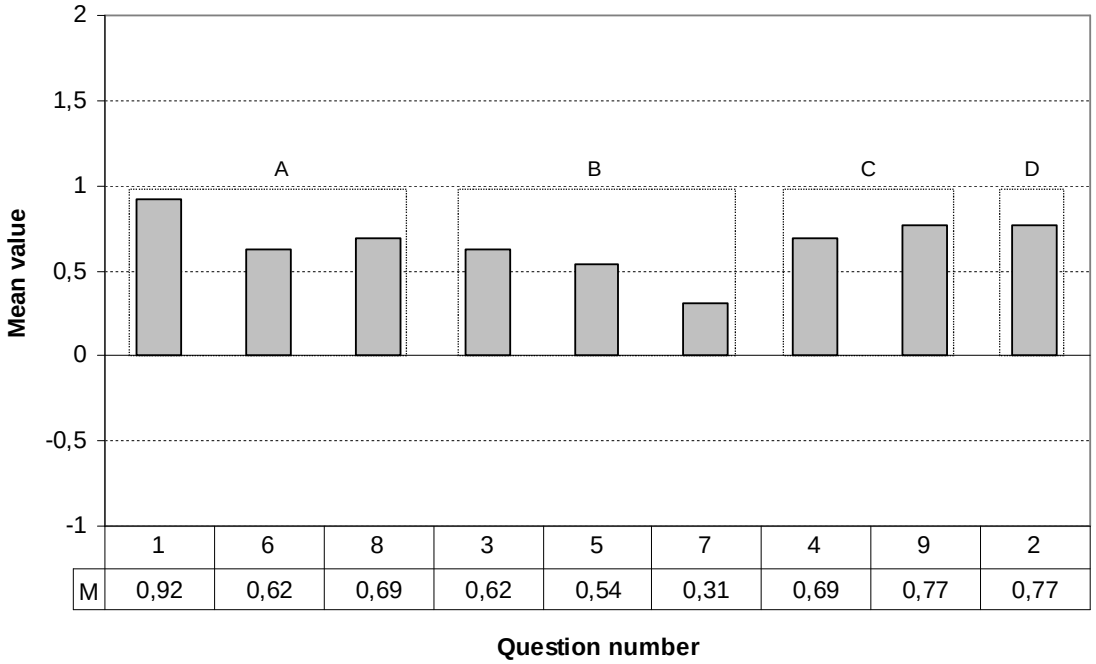


Figure 9.9: Comparative overview of usefulness estimation for **task patterns** according to question clusters - question numbers with mean value for each question are given on the x-axis

Ease of Use

The ease of use estimation for task patterns is not trivial as the concept is supported through functionality that is shared between different components. Task pattern extraction is initiated from the to-do list. Task pattern editing takes place in the Task Pattern Explorer. Task pattern

reuse is initiated in the to-do list, but task patterns are searched for and chosen for application in the Task Pattern Explorer. As the majority of the task-pattern related functionality is provided in the Task Pattern Explorer, the ease of use questions were focused on it.

Only 5 users had engaged with extracting and creating task patterns – 4 in TXTL and one in ASPL. Thus the ease of use evaluation is based only on 5 responses. The results are provided in Table D-8 and show positive ease of use perception, where negatively formulated questions are met with disagreement and positively formulated questions have received positive answers. The results point at the feasibility of direct manipulation techniques [Bla06], as these are utilized in the Task Pattern Explorer, for adaptation of weakly-structured task and process models for ad-hoc process support. Assessment of differences in the ease of use estimations between persons that were involved respectively in short-term and long-term CTM usage are not considered as only one person who used CTM for a short-term had worked with the Task Pattern Explorer.

9.4.2.1.5 Task Evolution

The concept of establishing evolutionary ancestor/descendant relationships between ad-hoc tasks for enabling interrelation and structured comparison between running processes and best-practices (R5) and between various best-practice variations (R6) has been evaluated based on the functionality for task evolution tracing provided in the CTM Task Evolution Explorer (cf. Figure 8.6). None of the users had engaged spontaneously in task evolution analysis. Therefore, only usefulness estimation based on the user experience from the CTM introduction and tutorials was possible. Ease of use evaluation is not provided for this component.

Usefulness

Usefulness estimation results are provided in Table D-9. Although users had not engaged with the task evolution functionality, this functionality was introduced to them in details during the CTM tutorials and exercises. 13 valid responses for the usefulness assessment of task evolution tracing were received. Task evolution tracing received the lowest usefulness estimation with an overall usefulness mean score of 0.08. The low overall score could be explained through the fact that the evolution tracing functionality strongly focuses on case analysis and optimization of best-practices, whereas the analysis of the accompanying qualitative interviews revealed that such case analysis and best-practice optimization were not a typical job for any of the study participants. Furthermore, ancestor/descendant relationships are established during reuse of task patterns. The restricted usage of task patterns thus impact negatively also on the usefulness estimations for task evolution analysis.

The inferential statistics further do not provide any evidence for different usefulness estimations between persons that have used CTM for a short-term period and those that were involved in long-term CTM usage (cf. Table D-11). No correlation between the number of managed persons and the usefulness estimation of task evolution tracing is further evident (cf. Table D-13).

Comparative Usefulness Cluster Overview

A comparative overview of the usefulness estimations for the task evolution tracing according to the different TAM question clusters (cf. Table 9.2) is shown in Figure 9.10. The results show that usefulness is not expected for any of the shown aspects except *control* over the job (cluster D).

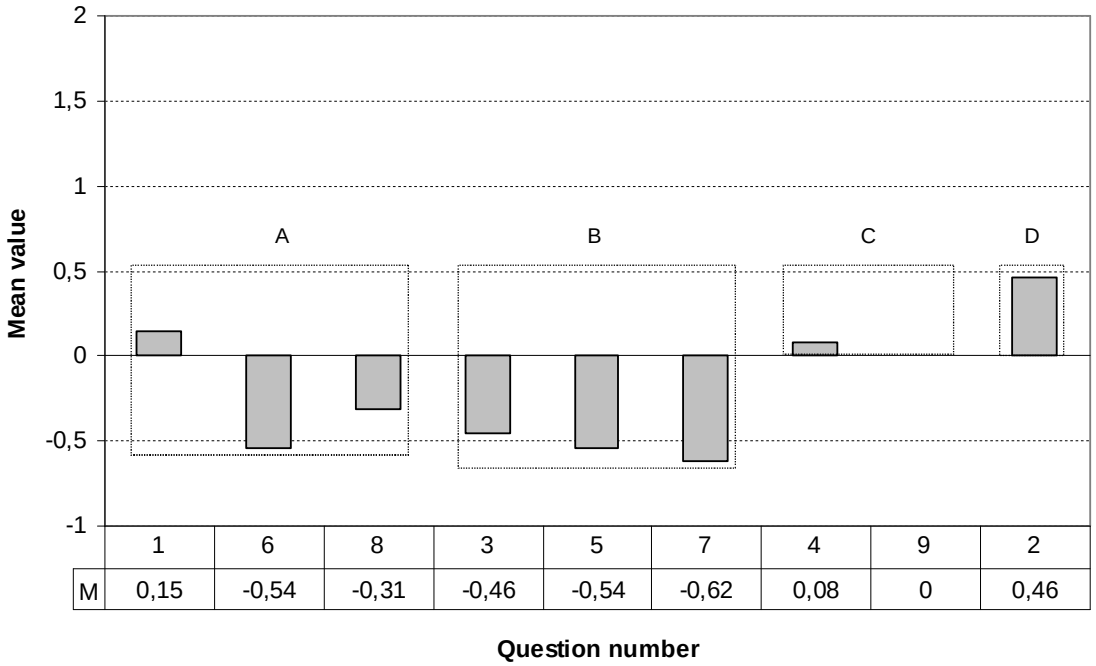


Figure 9.10: Comparative overview of usefulness estimation for **task evolution tracing** according to question clusters - question numbers with mean value for each question are given on the x-axis

9.4.2.1.6 Workflow Generation

The concept of supporting composition of structured workflow models for process automation (R7) was evaluated based on the CTM Workflow Editor. None of the users had engaged spontaneously in process transformation or formal workflow modeling. The discussed usefulness estimation is based on the user experience from the introduction and tutorials, and in TXTL also on the user participation in the process formalization studies (cf. Section 9.3.3). No ease of use estimation is provided for workflow model derivation as only ITL and ITE practically used the CTM Workflow Editor (cf. Section 9.3.3.1).

Usefulness

Usefulness estimation results are provided in Table D-10. The functionality for derivation of structured workflow models was introduced to the end users in details during the CTM introduction and tutorials. All users had filled the related usefulness questions, delivering 13 valid responses. Workflow generation received a positive overall usefulness mean score of 0.62. A MW test did not reveal any difference in the usefulness estimation between users engaged respectively in short-term and long-term CTM usage ($U=19$, $N_1=6$, $N_2=7$, $p_{(exact)}=0.836$, two-tailed, cf. Table D-11). This observation urged further analysis, as the group of users involved in long-term evaluation was expected to be better aware of the implications from workflow generation than those users that had used CTM only for a short period of time.

The lack of difference in the usefulness estimation could be explained later on during the qualitative interview analysis and through further inferential statistic. Eventually, users exposed to long-term CTM usage who participated also in the process transformation workshops (Section 9.3.3) reported that they did not intend to derive themselves or use formal workflows in long-term perspective, because of the lacking support for the jBPM functionality and the resulting additional overhead for the IT department (limitations for the evaluation are summarized later on in this

chapter). Thus, users considered workflow generation as a way to analyze processes rather than to automate them. Analysis was reported useful for already completed procedures to estimate how long a task has been handled in a given department based on the task change history, and in what sequence tasks have been performed. Such analysis was reported especially useful by CSE from ASPL who wanted to trace delays in customer order processing. ITL further stated that he would rather use workflow derivation to capture the status quo in running processes, i.e. to see if current ad-hoc tasks are performed in an optimal manner (in parallel), and to estimate the scope of undo procedures on related tasks. Post-process analysis was considered less useful by ITL as it would not allow him to correct a running process.

The fact that workflow generation was considered as analytical capability is confirmed through a positive Spearman correlation between the overall usefulness scores of workflow generation and task evolution tracing ($r_s = 0.6$, $p = 0.03$, two-tailed). The latter functionality clearly targeted at task and case analysis and best-practice consolidation.

Comparative Usefulness Cluster Overview

A comparative overview of the usefulness estimations for workflow generation according to the different TAM question clusters (cf. Table 9.2) is shown in Figure 9.11.

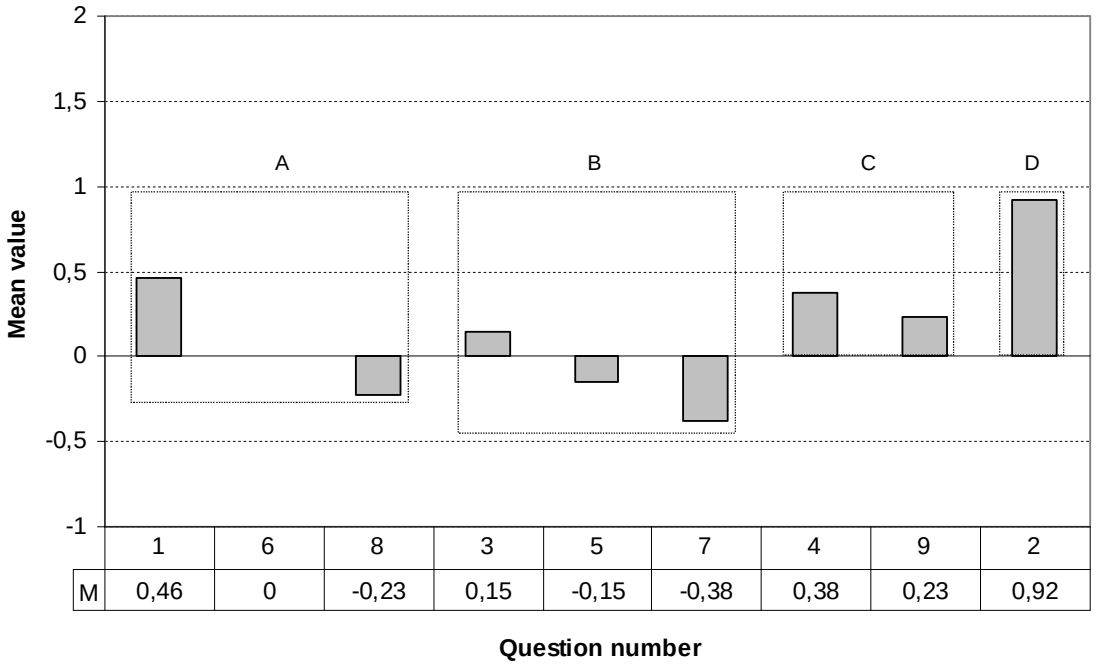


Figure 9.11: Comparative overview of usefulness estimation for **workflow generation** according to question clusters - question numbers with mean value for each question are given on the x-axis

Job effectiveness (cluster A) estimations reveal controversial usefulness expectations for workflow generation. While users estimate that the quality of work (question 1) may increase, there are no expectations that the effectiveness would increase (question 8). Similar results are available also for the task evolution tracing (cf. Figure 9.10). The results can be interpreted as implication that quality of work can be related to the analytical capabilities of task management applications whereas individual effectiveness does not relate to task and process analysis.

Job productivity and time savings (cluster B) is an aspect where no usefulness of workflow generation is expected overall.

Importance of the system to the users' job (cluster C) is an aspect for which positive usefulness results were received for workflow generation. The interviews revealed that analysis of actual tasks' flow and processing times (i.e. who does what, how long) are considered especially useful by end users.

Control over the job (cluster D) is the aspect with the highest positive usefulness estimation for workflow generation. The interviews showed that this aspect is also strongly influenced by the anticipated capabilities for process analysis.

The overall usefulness estimations for workflow generation are higher than those for task evolution tracing. This difference in the scores for both concepts can be explained through the fact that workflow generation allows analysis of ad-hoc process instances, whereas the analytical capabilities of task evolution tracing strongly depend on the definition and reuse of task patterns. During CTM test usage, the users were not able to develop strategies for task pattern management. The restricted usage of task patterns impacted negatively also on the usefulness estimations for task evolution tracing.

9.4.2.2 Estimations of Self-Efficacy, Benefits and Drawbacks

An overall assessment of the individual attitude of end users' towards engaging in process composition as well as on perceived possible benefits and drawbacks was enquired in the final section of the questionnaire. The questions were designed based on the self-efficacy, benefits, and drawbacks questions from the questionnaire from preliminary empirical studies (cf. Chapter 2). The questions and the results are shown in Table 9.3.

Table 9.3: Estimation of self-efficacy, benefits, and drawbacks for end-user driven business process composition

Self-efficacy	**M**	**SD**
Q1. I believe I am able to successfully describe and structure processes given just a software manual or a help facility, without anyone showing me what I should do	0,77	0,44
Q2. I believe I am able to successfully describe and structure processes provided I can call someone for help if I get stuck.	1	0,91
Benefits		
The process description developed by me could improve my effectiveness at work.	0,38	0,51
The process descriptions developed by me can provide me with better information about with better information about tasks and deadlines than the present software (like e.g. Email and MS Word documents).	0,77	0,60
Drawbacks		
If I was good at developing process descriptions, this may sidetrack me from my main career and result in a missed promotion because management would need my development skills where I am at present.	-1,31	0,63
If I made a mistake whilst developing process descriptions, I will loose credibility and esteem in the office.	-1,31	1,11
If I leave the company my process descriptions will not be understood by my colleagues.	-1,38	0,77
The time I spent for developing process descriptions was too long. / The time I spend for developing process descriptions will be too long.	-0,69	1,03
The time I spend developing process definitions will be greater than the time saved by me and others once these are developed.	-0,54	0,88

Positive self-efficacy and benefit estimations are visible in Table 9.3 with positive mean estimation scores. Drawbacks are met with disagreement. The results show that users have positive expectations of being able to participate in process composition based on personal task

management after having reviewed the proposed technical implementation – the CTM prototype.

9.4.3 Summary of Findings – TAM-Based Evaluation

The TAM-based evaluation has delivered positive usefulness estimations for all components of the CTM system, thus indicating feasibility of the underlying concepts. Figure 9.12 provides a comparative overview the overall usefulness mean scores for the different components.

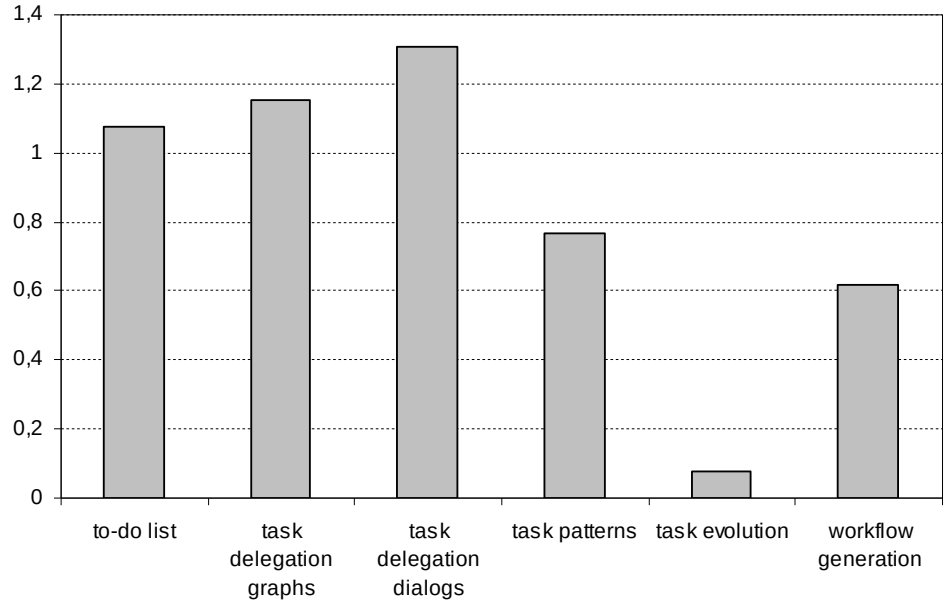


Figure 9.12: Comparative overview of overall usefulness estimations based on mean values

The **overviews provided by task delegation dialogs and task delegation graphs** have the highest overall usefulness mean scores. Thus, the need for transparency and structure in ad-hoc processes emerge as the strongest motivating factors for end-user driven process composition.

Task management in a light-weight to-do list has the third highest overall usefulness mean score. To-do lists are considered useful for providing ad-hoc planning and reminders for tasks.

Task patterns for extraction, adaptation and reuse of process knowledge are considered useful primarily by employees who have repeated (operational) tasks.

Task evolution tracing for the analysis of differences between running processes and best-practices and between different best-practice variations has the lowest usefulness score. The provided task evolution capabilities strongly depend on the establishment of task patterns as a central mean for best-practices definition. Restricted task pattern usage impacts on the usefulness estimations of task evolution tracing.

Workflow generation has a positive usefulness score. Workflow generation is considered as a technique for analysis of running and completed process instances. The analysis based on workflow generation does not depend on task patterns. Thus workflow generation has a higher usefulness score than the usefulness score of the task evolution tracing, which is also an analytical functionality.

Control of the users' job is the primary aspect where users perceive usefulness from all introduced concepts. Productivity is the aspect for which users have the lowest usefulness expectations. Thus, the provided task management and process management concepts and related components are seen as means to organize and manage activities, rather than to perform them.

Positive self-efficacy and benefit expectations for participating in process composition have been expressed by the end users after they have considered the provided set of components and

underlying concepts. Potential drawbacks from end-user driven business process composition are met with disagreement. Thus, the provided set of concepts adequately addresses involvement of end users in business process composition.

9.5 Limitations of the Evaluation

The evaluation of the developed concepts faced some limitations, which are summarized in the following to clarify the lack of scalability assessments especially for task pattern management.

Prototype usage in a restricted user group for a limited time-frame was only possible because of administrative restrictions in the research project. The prototype needed to be uninstalled at the partner companies after the end of the research project. Thereby the companies needed to switch back to their previous practice. As a result the industrial partners were not motivated to spread out the prototype and to adjust their work practices to the provided system.

The industrial partners had already central document management systems, i.e. in TXTL such a system was based on shared file folders, in SWVP document management was performed through an internal web-portal, and in ASPL various shared folders and a web-portal were used. Therefore the implementation of a proprietary document management system for evaluating externally-managed artifacts in the CTM prototype was not feasible, as it could not replace the existing document management systems in the evaluation target companies. On the other hand, integration of the respective partner systems for document management in the CTM prototype was not possible due to resource limitations.

A further consequence from the limited time frame for CTM system usage at the application partners was that users did not invest extensive efforts in composing task patterns, weakly-structured or structured process models but rather experimented to evaluate what potential benefits could arise for their company from the provided system. This attitude did not allow the extensive generation, distribution, and reuse of task patterns. The restricted task pattern usage affected negatively also the use of the task evolution tracing, as evolutionary relationships rely on the extraction and reuse of task patterns.

No long term support for the system or for related technologies like jBPM was further provided. Even if jBPM workflows could be developed to streamline given processes, the related technologies were seen as a possible additional overhead for the IT departments. The IT departments in the partner companies considered that they may not be capable of maintaining the developed process models and the underlying infrastructure if problems occur. This hindered the uptake of the technologies in long-term perspective.

9.6 Summary

This chapter has presented the evaluation of the developed concepts for end-user driven business process composition. Several evaluation phases have been discussed.

A short-term qualitative evaluation phase has been presented, during which preliminary user feedback on the developed concepts and their realization in the CTM system has been collected. The preliminary qualitative assessment of the underlying concepts delivers positive results and a set of user proposed extensions considering lower level concepts and implementation details. These extensions have been partially implemented in an extended version of the CTM system.

A long term evaluation phase has been further presented. This evaluation phase has delivered a set of captured ad-hoc process instances. The captured processes have revealed details about users' strategies for managing ad-hoc work and about the level of specificity of ad-hoc task delegation graphs. The evaluation results have shown that the proposed concepts are feasible for

top-down business process composition. Concept refinements have been proposed to deal with inconsistencies resulting from bottom-up extensions of task delegation graphs.

A series of qualitative case studies have been further conducted to evaluate the transformation of ad-hoc task delegation graphs to structured workflow models. The focus in the different case studies was respectively: on process tailoring by local developers, on process transformation through involvement of end users, and on workflow model refinement based on ad-hoc task deviations from workflow instances. The results have shown that the provided concepts are feasible for end-user driven composition of structured process models. The need to support transformation and validation of formal workflow models through business technology experts (process designers or developers) has been perceived.

A quantitative evaluation of the process composition concepts based on the TAM [Dav85, Dav89] has been further discussed. This evaluation has focused on the major concepts for end-user driven business process composition through referring to the corresponding CTM components. The positive usefulness assessments for the different components have shown that the developed concepts are adequate for involving end users in business process composition. The primary aspect where usefulness has been perceived is control over the users' job. The motivating factors for involving end users in business process composition can be summarized as enhanced personal task and information management including planning and reminders for to-do items, and transparency, structure and analysis of human-centric, collaborative processes. The detected positive intent of end users towards engaging in process tailoring activities from the TAM-focused assessments is supported through positive self-efficacy assessments and a suitable balance between drawback and benefit expectations of end users with respect to business process composition based on collaborative task management.

CHAPTER 10: Conclusions and Future Work

This chapter provides conclusions on the research topics covered in the thesis. A summary of the contributions is further provided which stresses on the most important aspects of the developed concepts. The chapter further discusses implications from the presented work for the areas of business process management and end-user development. The chapter concludes with an outlook for future research directions.

10.1 Conclusions

The thesis provides a conceptual framework for enabling end-user driven business process composition. With respect to weakly-structured, knowledge-intensive processes, the thesis addresses the novel paradigm of the *"Process of Me"* [Gar06]. This paradigm propagates a need to support the individual's perspective on business processes and to leverage the individual's process knowledge towards the achievement of common enterprise goals. The necessity for user-centric process support is further leveraged in research literature that discusses enterprise efficiency as a result from the individual actions of all employees [Wii04].

A particular challenge for next generation business process management systems in the area of operational process support is seen in the need to involve end users in the composition of formal process models. This need relates to reconciling the domain knowledge of business users and the business technology knowledge of process designers and developers in the course of workflow projects [For06]. The need to provide a shared context for process tailoring between end users and developers is recognized also in research literature [MM00]. Thus, through addressing end-user driven business process composition in the complete spectrum, form ad-hoc to procedural process support, the thesis addresses research challenges, which are perceived as equally relevant from industrial as well as from research literature. The discussed challenges are addressed from an end-user development [LPKW06] perspective.

The developed concepts are based on preliminary empirical studies, which have been conducted in two different directions. A questionnaire-based survey has provided assessment of current end-users work practices, attitudes, and intentions related to end-user development in the domain of task management [MSG+08]. Task management has been selected especially because of the need to bridge the individual, task management perspective with the enterprise business process management perspective [RRMvdA05]. The results from the survey have provided strong support for applying end-user development techniques in the domain of task management.

A set of field-studies have been conducted at industrial partner companies to assess current end user problems related to task management and to the management of informal business processes. The discovered problems have revealed entry points for introducing process tailoring by end users. The results from the empirical studies have been used to elaborate a set of generic requirements for end-user driven business process composition.

A detailed state of the art analysis has been presented, based on the elaborated requirements from the empirical studies. The state of the art analysis has been performed by considering related research in two major directions – end-user development and user-centric process support.

The state of the art analysis in the domain of end-user development has delivered a set of intrinsic concepts related to tailoring of software artifacts by end users. These concepts have been considered as basic meta-approaches for end-user driven business process composition and mapped to the requirements from the preliminary empirical studies. Analysis of available end-users development approaches has been further provided, which has focused on different possibilities to realize process composition by end users. Programming by example [Cyp93,

Lie01] has been identified as the most appropriate approach, addressing end users without technical skills or process modeling knowledge.

The state of the art analysis focusing on user-centric process support has discussed process composition approaches from a variety of research fields. Various approaches and notations for structured process modeling have been discussed. Process mining has been considered as an unobtrusive composition technique which however does not provide sufficient control to the end users for changing emerging process definitions. Various approaches from the domains of knowledge management and computer-supported cooperative work have been further discussed. All approaches have been discussed in the context of the elaborated requirements from the empirical studies and the fundamental end-user development concepts to clarify the research gap and the need for the concepts presented in the thesis.

A set of concepts have been elaborated for enabling end-user driven business process composition. These concepts enable enterprise-wide programming by example based on collaborative, email-integrated task management. Through the tight integration in the actual working environment and the flexible composition and adaptation of user-defined process models, the introduced concepts provide a gentle slope of complexity [MCLM90] for process tailoring by end users. A reduced expertise tension [Ber94] during process composition by end users is enabled through enhanced guidance based on implicitly captured previous process knowledge or explicit best-practices. Iterative refinement of user-defined process models is enabled through the possibility for extraction, exchange, adaptation, and reuse of user-defined task hierarchies which support the *"seeding, evolutionary growth, and reseeding (SER)"* process model [FGY+04]. Process tailoring as collaboration [MM00] is enabled through a seamless transition and a shared context between user-defined and formal process models.

The concepts introduced in the thesis enable informed participation of end users in business process composition by introducing several gentle slopes of complexity and providing added value on personal task management as motivation to overcome each one of them. The implementation and validation of the concepts through the developed CTM prototype system has shown that the chosen approach is feasible for enabling end-user driven business process composition. Personal task management and exchange of tasks and deliverables over email enable light-weight support for managing and organizing individual's activities according to existing end users' work practices. Through implicitly generated task delegation graphs users are enabled to get transparency in collaborative processes, exceeding the capabilities of common email and to-do list applications. Task related dialogs further provide structured overview of task-related email exchange and reduce search efforts. Task patterns enable users to establish best-practices and to exchange, adapt, and reuse previous experience. The transformation of ad-hoc processes to formal workflows enables process analysis and provides common ground for consolidation of enterprise processes between business users, process designers and developers.

The evaluation studies have shown that the organizational culture and user motivation are vital for the acceptance of end-user driven process composition. Pitfalls and risks hide not only in the different attitude of end users towards using formal systems and maintaining task information on regular basis, but also in the ability, expertise, and willingness of local developers and management to support end-user driven business process composition in long-term perspective. The introduction of new technologies can result in additional overhead for local developers and IT departments. Shifting the focus of the latter parties from the common daily work to supporting end users in process composition can raise a barrier for the uptake of end-user driven process composition in enterprises. Thus techniques such as direct manipulation and programming by example need to be considered, where task and process representations are self-explanatory and where end users can cope with the process technology to an advanced level.

10.2 Contributions

The thesis has introduced several major concepts for enabling end-user driven business process composition, which constitute the scientific contributions of the thesis.

A **task management model** has been presented, which enables aggregation of data from the underlying task management (to-do list) and collaborative (email) applications to construct weakly-structured, user-defined process models [SSS07]. The task management model comprises two major parts – a runtime task management model and a task pattern model.

The *runtime task management model* defines the logical entities and relationships that are used to capture data in running ad-hoc process instances. The attributes for the different entities have been further provided to clarify the context information of task instances. The association of different entities for control flow (tasks), document flow (artifacts), and human actors (users) has been described. Changes of the various entity types have been further introduced in the model. The runtime task management model has further defined basic associations between ad-hoc task instances and workflow task models as well as between workflow task models and workflow task instances. These associations support derivation of structured workflow models from user-defined, ad-hoc task hierarchies and refinement of workflow models based on deviations from workflow instances through ad-hoc tasks.

The *task pattern model* defines reusable task models for ad-hoc task instances. Task patterns represent explicit best-practices that are decoupled from a concrete process instance. Task patterns can be extracted from ad-hoc process instances. During task pattern extraction, runtime information such as associations to dialogs and messages for exchange of tasks and deliverables is removed from the generated task pattern. Through this, task pattern abstracts from a specific handling of a business case towards a generic best-practice definition. Task patterns can be further defined from scratch as explicit best-practices for ad-hoc processes. Evolutionary ancestor/descendant relationships are maintained to enable enhanced task analysis based on the reuse of task patterns.

A **method for composition of weakly-structured process models** has been further introduced [SSS07]. It defines how emerging ad-hoc task instances are aggregated to overall task delegation graphs. Task delegation graphs are constructed through tracking of user activities on task instances from the local workspace and replicating user-defined task hierarchies on a central server instance. Task hierarchies of multiple users are bound to an overall task delegation graph based on the tracked email exchange for task delegation. Thus, task delegation graphs are captured process execution examples, which are developed by different process participants where each process participant manages and structures (models) the emerging ad-hoc process in their area of expertise. The introduced method for composition of weakly-structured process models defines the collaborative task handling and the binding of email messages to task delegation dialogs. The method further defines local and global scopes in task delegation graphs and discusses critical aspects related to changes that affect the tasks of multiple users, i.e. changes in the global scope. The transitions between ad-hoc task instances and task patterns and vice versa have been further discussed in the provided method. Local and global scopes for task patterns have been introduced for different spectra of task pattern visibility and reuse. Critical aspects of task pattern editing and data transfer between local and global task pattern scopes have been further addressed in the presented method. The method has further defined the mechanisms for establishing ancestor/descendant relationships resulting from task pattern reuse. These relationships facilitate task analysis in the context of SER [FGY+04].

A **method for composition of structured process models** has been introduced [SSFM08d]. It enables the transformation of weakly-structured task delegation graphs to structured workflow models. The method has provided different interpretations of hierarchical task decomposition and delegation flow. The control flow in a derived workflow model is based on the structural

relationships between ad-hoc task instances in a task delegation graph and on the change history of the ad-hoc task instances. The method has discussed possibilities for selection of different task change types from the history for précising the derived process models. Task ranges are constructed from the captured task changes. The temporal relationships between task ranges determine also the temporal relationships between the task nodes (workflow task models) in a derived formal workflow model. Discrepancies between task ranges and workflow graph correctness criteria have been discussed and a mechanism for resolving these discrepancies through different consolidation options. The method for composition of structured process models has further defined how the relationships between ad-hoc task instances and workflow task models (task nodes) resulting from deviation are used to enable extensions of formal workflow models based on user-defined ad-hoc task hierarchies.

A **holistic concept for end-user driven business process composition** has been further introduced, which composes the task management model and the process composition methods into a seamless overarching method and architecture for the composition of weakly-structured and structured process models [SSFM08b, SS08]. The overarching method considers gradual user involvement in business process composition by exposing different functional system components to the end users. Involvement starts on personal task management level in the local end users' workspace through management of light-weight, personal to-do lists and through task delegation over email. Further involvement is motivated through enhanced transparency and analysis of running collaborative processes provided through task delegation graphs and dialogs. Users that are interested in best-practice extraction, adaptation, and reuse are provided with an environment for task pattern management. Finally, enhanced task and process analysis is enabled through tracing of task evolution and formalization of captured ad-hoc processes. Thus, through the various provided functionalities for personal task management, ad-hoc, and procedural process support, the holistic concept bridges the gaps between personal task management and business process management [Gar06, RRMvdA05], and between the end users', process designers', and developers' perspectives on business processes [For06, MM00].

10.3 Implications for Business Process Management

The concepts introduced in the thesis provide a solid ground for developing user-centered process composition systems. Such systems can facilitate the adoption of business process management technology by reducing the time and costs for implementation of workflow projects in enterprises. Particularly, the developed concepts show how the *"Process of Me"* [Gar06] can be implemented in a top-down manner through reconciling data on personal task management of multiple process participants towards the generation of end-to-end, weakly-structured process models. A particular challenge remains to explore techniques that can increase the user motivation to contribute to the composition of such process models at an increased level of details towards the generation of comprehensive enterprise process knowledge. Such techniques need to consider the tradeoff between cognitive effort for the contribution and benefits that result from process knowledge reuse [RRMvdA05]. Additionally, privacy aspects need to be considered for end-user driven business process composition in large organizational settings. Approaches for managing of user-defined, weakly-structured process models need to be additionally considered to ensure that these models will scale during long-term adaptation and reuse. Such scalability may require generic or enterprise-specific policies for auditing, publishing, and accessing user-defined process models.

The concepts introduced in the thesis further enable the derivation of structured workflow models from user-defined, weakly-structured process models. The developed approach enables "*increased collaboration in process modeling*" [For06] through a shared context between user-

defined and formal process models. Furthermore, end users are enabled to refine initial workflow models through deviations with ad-hoc task hierarchies at runtime. Thus end-user driven process composition is enabled from process discovery, to process design and redesign [Ver04]. A major challenge remains to develop management approaches for the enabled process tailoring as collaboration [MM00] between end users, process designers, and developers. Such tailoring can require clearly defined responsibility areas and user roles to keep business processes consistent and reliable. For example, deployment and instantiation of user-defined workflow models prior to their validation and consolidation with technical experts can result in errors and additional costs.

10.4 Implications for End-User Development

Business processes can require domain specific knowledge of multiple end users, who are responsible for different process areas. Therefore the composition of business process models cannot be addressed through conventional end-user development approaches, focusing on the development of software artifacts in a single application environment or user workspace. The thesis has introduced the generic concept of collaborative programming by example [Cyp93, Lie01] of business process models as reusable software artifacts on enterprise level. The concept is realized through enabling each user to tailor the process facets in their area of expertise, i.e. according to the individual domain knowledge and problem solving strategies, and integrating captured data from the individual process tailoring environments of multiple users into a common software artifact, i.e. an emergent process model. An overview of the collaboratively developed, emergent process model is provided to enable *"social creativity in which all stakeholders reach a shared understanding by contributing their different points of view and knowledge"* [FGY+04].

Collaborative tailoring is performed by considering different user attitudes towards managing process data and different domain and technical expertise of end users. The complexity of process tailoring is separated in different functional components. Personal to-do list and email are accessible to end users without any or with few technical skills. Navigation in the overall process overview, and extraction, adaptation, and reuse of task patterns can be performed by end users and local developers who need respectively transparency and reuse of previous process knowledge. For supporting gradual involvement of end users in tailoring on enterprise level, appropriate modular architectures need to be considered that address different aspects of users' work and different information needs of end users. For example, the evaluation of the developed process composition concepts has shown that data and process analysis are needed by end users in many situations in their daily work. Thus, analytical capabilities in business applications can be used to motivate end users to engage in tailoring of software artifact through direct manipulation, e.g. through explicitly recording and planning activities or through tagging information artifacts.

Business processes are influenced not only by the technical realization of the enterprise systems but also by the organizational structure of enterprises. [WR95] suggests that *"the relationship between the technical and the organizational changes is characterized by reciprocity and interdependence"*. Hence, the adoption of end-user driven business process composition in enterprises can be realized only if process composition environments account for the social context in organizations. For example, publishing and reuse of misleading best-practices that hold outdated or irrelevant process knowledge can bring tension and harm the organization. Therefore as particular challenge remains to develop management principles for dealing with changes in user-defined process models and for keeping the overall socio-technical environment trustworthy and efficient. Particularly, comprehensive models and assessment criteria for end-user development for business process composition are needed that are able to take into account technological factors as well as benefits, drawbacks, costs, risks, and other factors that affect the uptake of end-user driven business process composition on personal and on organizational level.

10.5 Future Directions

The thesis has provided mechanisms for involving end users in business process composition, by revealing further challenges and open research questions. These questions relate particularly to task pattern management and the evolutionary relationships between reused task hierarchies. Strategies for task pattern management are of particular interest, as task patters are reusable fragments for ad-hoc processes that can be developed and edited by end users through direct manipulation techniques [Bla06] without requiring any knowledge of a formal modeling language. Scalability assessments of task pattern management and task evolution tracing were not possible in the presented research because of the restricted usage of the developed prototype system. Task pattern reuse can be observed and evaluated only in long term observations, where task patterns are the rule for best-practice definitions. Future research initiatives will aim at the application of the CTM system and the underlying concepts in real-life enterprise settings towards observing and evaluating task pattern management strategies and task evolution.

Further, the thesis has focused on light-weight composition of business process models based on personal task management. A major focus thereby has been set on possibly reducing the cognitive burden of specifying explicit constraints and formal dependencies between tasks in ad-hoc processes. However, defining formal constraints may be needed to model the interactive systems' behavior in organizational settings by configuring reusable task models (task patterns) and emerging processes (task delegation graphs). For example, it may be needed to specify which users should be able to see which tasks in a task delegation graph. It may be further considered enabling users that are not directly involved in an ad-hoc process to contribute to some tasks by providing comments, attaching documents to tasks, or escalating critical tasks in task delegation graphs. In this case task delegation graphs will serve as a collaborative workspace where input from the personal task hierarchies can be enriched and consolidated by various users. First steps in this direction have been reported in [SS08b]. The latter study reports work in progress which has not been included in the thesis. A further related aspect that needs to be considered for using task delegation graphs as collaborative, shared accessible workspaces is the flow of process data from task delegation graphs back to the ad-hoc task hierarchies in the individual user workspaces.

The concepts introduced in the thesis further address a seamless transition between user-defined, weakly-structured process models and structured workflow models. The trends of computer-supported cooperative work and business process management approach such transition from different perspectives. While computer-supported cooperative work studies consider embedding structure in ad-hoc operations [Bar01], business process management studies focus on embedding ad-hoc human tasks into structured processes. The latter aspect is in the focus of current initiatives for developing new human task standards for formal process definition languages such as the WS-human task standard [AAD+07]. Transformation of ad-hoc processes or process fragments in the form of task hierarchies or task delegation graphs to formal workflow modeling notations is not addressed in scientific and industrial research. Considering the ongoing standardization efforts for bridging operational and ad-hoc processes, the transformations discussed in the thesis can provide valuable extension to known process composition approaches.

A particular research challenge remains to evaluate transformation of ad-hoc task structures to workflow models by considering evolutionary relationships from similar task pattern application cases. Such transformations can enrich the derived task flow with alternative flows. Further research is needed to asses whether loops can be derived from collaborative actions on ad-hoc task instances in task delegation graphs, such as e.g. declination and approval of completion declarations and reopening of ad-hoc task instances.

Finally, the thesis has approached end-user driven business process composition from end

users' perspective. However, rendering the composition and appropriation of process models to end users can bring potential threats from business technology perspective. For example, it may be not be desired to enable tailoring of business process models by end users in all aspects in order to ensure that these models are developed according to the correct formal constraints. Therefore the feasibility of end-user driven business process composition from the perspective of business technology experts and developers needs to be further investigated.

Bibliography

[ADMG97] A. Agostini, G. De Michelis, and M. A. Grasso. Rethinking CSCW systems: the architecture of Milano. In: *Proceedings of the European Conference on Computer Supported Cooperative Work (ECSCW)*, pp. 33-48. Springer, 1997.

[ADML03] A. Agostini, G. De Michelis, and M. Loregian. Undo in Workflow Management Systems. In: W.M.P. van der Aalst et al. (Eds.): *BPM 2003, LNCS 2678*, pp. 321–335. Springer Berlin Heidelberg, 2003.

[AAD+07] A. Agrawal, M. Amend, M. Das, M. Ford, C. Keller, M. Kloppmann, D. König, F. Leymann, R. Müller, G. Pfau, K. Plösser, R. Rangaswamy, A. Rickayzen, M. Rowley, P. Schmidt, I. Trickovic, A. Yiu, and M. Zeller. *Web Services Human Task (WS-HumanTask)*, Version 1.0, June 2007. URL http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel4people/WS-HumanTask_v1.pdf [27.01.2009]

[Ajz85] I. Ajzen. From intentions to actions: A theory of planned behavior. In: J. Kuhl & J. Beckman (Eds.), *Action-control: From cognition to behavior*, pp. 11-39. Springer, Heidelberg, 1985.

[ALL88] E. Auramäki, E. Lehtinen, and K. Lyytinen. A Speech-Act-Based Office Modeling Approach. In: *ACM Transactions on Information Systems (TOIS)*, vol. 6, no. 2 (April 1988), pp. 126-152. ACM, New York, 1988.

[Ann04] J. Annett. Hierarchical Task Analysis. In: Dan Diaper and Neville Stanton (Eds.): *The Handbook of Task Analysis for Human-Computer Interaction*, pp. 67-82. Lawrence Erlbaum Associates, 2004.

[AS00] J. Annett and N. Stanton. *Task analysis*. Taylor & Francis, 2000.

[AS94] K. R. Abbott and S. K. Sarin. Experiences with Workflow Management: Issues for the Next Generation. In: *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pp. 113–120. ACM Press, New York, 1994.

[Bar97] J. E. Bardram. Plans as Situated Action: An Activity Theory Approach to Workflow Systems. In: *Proceedings of the European Conference on Computer Supported Cooperative Work (ECSCW)*, Lancaster, UK, pp. 17-32, 2004.

[BB93] J. C. Brancheau and C. V. Brown. The Management of End-User Computing: Status and Directions. *ACM Computing Surveys*, vol. 25, no. 4 (December 1993), pp. 437-482. ACM Press, 1993.

[BBCS08] C. Bogart, M. Burnett, A. Cypher, and C. Scaffidi. End-User Programming in the Wild: A Field Study of CoScripter Scripts. In: *Proceedings 2008 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Herrsching am Ammersee, Germany, pp. 39-46. IEEE Computer Society, 2008.

[BBPS06] J. E. Bardram, J. Bunde-Pedersen, and M. Soegaard. Support for activity-based computing in a personal computing operating system. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pp. 211-220. ACM Press, New York, 2006.

[BCR04] M. Burnett, C. Cook, and G. Rothermel. End-user software engineering. *Communications of the ACM*, vol. 47, no. 9 (September 2004), pp. 53-58. ACM Press, 2004.

[BD06] J. Bortz and N. Döring. *Forschungsmethoden und Evaluation für Human- und Sozialwissenschaftler*, 4. Auflage. Springer Medizin Verlag

[BDG+04] V. Bellotti, B. Dalal, N. Good, P. Flynn, D. G. Bobrow, and N. Ducheneaut. What a To-Do: Studies of Task Management towards the Design of a Personal Task List Manager. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pp. 735-742. ACM Press, New York, 2004.

[BDH+05] V. Bellotti, N. Ducheneaut, M. Howard, I. Smith, and R. Grinter. Quality Versus Quantity: E-Mail-Centric Task Management and Its Relation With Overload. *Human-Computer Interaction*, vol. 20 (2005), pp. 89-138. Lawrence Erlbaum Associates, 2005.

[BDHS03] V. Bellotti, N. Ducheneaut, M. Howard, and I. Smith. Taking Email to Task: The Design and Evaluation of a Task Management Centered Email Tool. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, Ft. Lauderdale, Florida, USA, pp. 345–352. ACM Press, New York, 2003.

[Ber05] P. Berens. The FLOWer Case-Handling Approach: Beyound Workflow Management. In: Marlon Dumas, Wil van der Aalst, Arthur ter Hofstede (Eds.): *Process-Aware Information Systems. Bridging People and Software through Process Technology*, pp. 363-395. John Wiley & Sons, 2005.

[Ber04] J. Beringer. Reducing expertise tension. *Communications of the ACM*, vol. 47, no. 9 (September 2004), pp. 39-40. ACM Press, 2004.

[Ber00] A. Bernstein. How Can Cooperative Work Tools Support Dynamic Group Processes? Bridging the Specificity Frontier. In: *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pp. 279-288. ACM Press, New York, 2000.

[BH98] H. Beyer and K. Holtzblatt. *Contextual Design: Defining Customer-Centered Systems*. Morgan Kaufmann, 1998.

[Bla02] A. Blackwell. First Steps in Programming: A Rationale for Attention Investment Models. In: *Proceedings of IEEE 2002 Symposium on Human Centric Computing Languages and Environments (HCC'02)*. IEEE Computer Society, 2002.

[Bla06] A. Blackwell. Psychological Issues in End-User Programming. In: Henry Lieberman, Fabio Paternó and Volker Wulf (Eds.): *End-User Development*, pp. 9-30. Springer, 2006.

[Bor92] N. S. Borenstein. Computational Mail as Network Infrastructure for Computer-Supported Cooperative Work. In: *Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW)*, pp. 67-74. ACM Press, New York, 1992.

[BRC06] M. Burnett, G. Rothermel, and C. Cook. An Integrated Software Engineering Approach for End-User Programmers. In: Henry Lieberman, Fabio Paternó and Volker Wulf (Eds.): *End-User Development*, pp. 87-113. Springer, 2006.

[But94] G. Button. What's Wrong With Speech-Act Theory. *Computer Supported Cooperative Work*, vol. 3, no.1 (Mar. 1994), pp. 39-42. Springer, 1994.

[CFMP06] M. F. Costabile, D. Fogli, P. Musso, and A. Piccinno. End-User Development: The Software Shaping Workshop Approach. In: Henry Lieberman, Fabio Paternó and Volker Wulf (Eds.): *End-User Development*, pp. 183-205. Springer, 2006.

[Cha08] *Chandler project*. URL http://chandlerproject.org/ [10.11.2008].

[CHH99] D. R. Compeau, C. A. Higgins, and S. L. Huff. Social Cognitive Theory and Individual Reactions to Computing Technology: A Longitudinal Study. *MIS Quarterly*, vol. 23, no. 2 (1999), pp. 145-158.

[CMN83] S. K. Card, T. P. Moran, and A. Newell. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates Inc., Hillsdale, 1983.

[Cyp93] A. Cypher. *Watch What I Do: Programming by Demonstration*. MIT Press, 1993.

[Dav85] F. D. Davis. *A Technology Acceptance Model for Empirically Testing New End-User Information Systems: Theory and Results*. Submitted to the Sloan School of Management in partial fulfillment of the Degree of Ph. D. in Management at the Massachusetts Institute of Technology, December 20, 1985.

[Dav89] F. D. Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, vol. 13, no. 3 (1989), pp. 319-340.

[DS90] T. H. Davenport and J. E. Short. The New Industrial Engineering: Information Technology and Business Process Redesign. *Sloan Management Review*, (Summer 1990), pp. 11-27. MIT, 1990.

[Dav98] T. Davenport. Putting the Enterprise into the Enterprise System. *Harvard Business Review*, vol. 76, no. 4 (July-August 1998), pp. 121-131. Harvard University Graduate School of Business Administration, 1998.

[DB01] N. Ducheneaut and V. Bellotti. E-mail as habitat: an exploration of embedded personal information management. In: *ACM interactions*, vol. 8, no. 5 (September/October 2001), pp. 30-38. ACM Press, New York, 2001.

[Des05] J. Desel. Process Modeling Using Petri Nets. In: Marlon Dumas, Wil van der Aalst, Arthur ter Hofstede (Eds.): *Process-Aware Information Systems. Bridging People and Software through Process Technology*, pp. 149-176. John Wiley & Sons, 2005.

[DHvdA05] S. Dustdar, T. Hoffmann, and W. M. P. van der Aalst. Mining of ad-hoc Business Processes with TeamLog. *Data and Knowledge Engineering*, vol. 55, no. 2 (November 2005), pp. 129-158. Elsevier B.V., 2005.

[Die00] R. Diestel. *Graph Theory*. Springer-Verlag, New York, 2000.

[DSB06] C. S. De Souza and S. D. J. Barbosa. A Semiotic Framing for End-User Development. In: Henry Lieberman, Fabio Paternó and Volker Wulf (Eds.): *End-User Development*, pp. 401-426. Springer, 2006.

[Dus04] S. Dustar. Caramba - A Process-Aware Collaboration System Supporting Ad hoc and Collaborative Processes in Virtual Teams. *Distributed and Parallel Databases*, 15, pp. 45–66. Kluwer Academic Publishers, 2004.

[DvdAtH05] M. Dumas, W. M. P. van der Aalst, A. H. M. ter Hofstede. *Process-Aware Information Systems. Bridging People and Software through Process Technology*. Marlon Dumas, Wil van der Aalst, Arthur ter Hofstede (Eds.). John Wiley & Sons, 2005.

[Ecl09] Eclipse org. URL http://www.eclipse.org/

[EFHT05] G. Engels, A. Förster, R. Heckel, and S. Thöne. Process Modeling Using UML. In: Marlon Dumas, Wil van der Aalst, Arthur ter Hofstede (Eds.): *Process-Aware Information Systems. Bridging People and Software through Process Technology*, pp. 85-117. John Wiley & Sons, 2005.

[Esk05] Esker Software. Esker Survey: Organizations Still Use Word Processors and Spreadsheets for Transactional Business Documents and Rely on Postal Mail for Delivery. URL http://www.esker.com/data/press_releases/news/PR_Office%20document%20survey_May05.pdf [13.11.2008].

[Esl08] I. Eslick. ScratchTalk and Social Computation: Towards a natural language scripting model. In: *Proceedings of the 2008 Workshop on Common Sense Knowledge and Goal-Oriented Interfaces (GSKGOI 2008)*, January 13, 2008, Canary Islands, Spain, 2008.

[EUD06] EUDISMES Project, 2006. URL http://www.eudismes.de/ [04.11.2008].

[FA75] M. Fishbein and I. Ajzen. *Belief, Attitude, Intention, and Behavior: An Introduction to Theory and Research*. Addison-Wesley, 1975.

[FGHW88] F. Flores, M. Graves, B. Hartfield, and T. Winograd. Computer systems and the design of organizational interaction. *ACM Transactions on Information Systems*, vol. 6, no. 2 (April 1988), pp. 153-172. ACM, New York, 1988.

[FGY+04] G. Fischer, E. Giaccardi, Y. Ye, A. Sutcliffe, and N. Mehanjiev. Meta-Design: A Manifesto for End-User Development. *Communications of the ACM*, vol. 47, no. 9 (Sep. 2004). ACM Press, 2004.

[For06] Forrester Research. *Increase Business Agility with BPM Suites*. Forrester Research Inc., 2006.

[Gad08] A. Gadatsch. *Grundkurs Geschäftsprozess-Management*. Friedr. Vieweg & Sohn Verlag, GWV Fachverlage GmbH, Wiesbaden, 2008.

[Gar06] Gartner Research. *Person-to-Process Interaction Emerges as the 'Process of Me'*. Gartner Inc., 2006.

[GH98] J. Grundy and J. Hosking. Serendipity: Integrated Environment Support for Process Modelling, Enactment and Work Coordination. *Automated Software Engineering: An International Journal*, vol. 5, no. 1 (Jan. 1998), pp. 27-60. Kluwer Academic Publishers, 1998.

[GN92] M. Gantt and B. Nardi. Gardeners and gurus: patterns of cooperation among CAD users. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, Monterey, California, USA, pp. 107-117. ACM Press, New York, 1992.

[GOR+07] O. Grebner, E. Ong, U. Riss, M. Brunzel, A. Bernardi, and T. Roth-Berghofer. *Task Management Model*. URL http://nepomuk.semanticdesktop.org/xwiki/bin/view/Main1/D3%2D1 [28.04.2009].

[Gra89] D. Graham. Incremental development: review of nonmonolithic life-cycle development models. *Information and Software Technology*, vol. 31, no. 1 (January/February 1989), pp. 7-20. Butterworth-Heinemann Newton, MA, USA, 1989.

[Gre89] T. R. G. Green. Cognitive dimensions of notations. In: A. Sutcliffe and L. Macaulay (Eds.) *People and Computers V*, pp. 443-460. Cambridge University Press, Cambridge, UK, 1989.

[GRM+04] D. Gruen, S. L. Rohall, S. Minassian, B. Kerr, P. Moody, B. Stachel, M. Wattenberg, and E. Wilcox. Lessons from the ReMail Prototypes. In: *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pp. 152–161. ACM Press, New York, 2004.

[GSSF04] A. Gaffar, D. Sinnig, A. Seffah, and P. Forbig. Modeling patterns for task models. In: *Proceedings of the 3rd annual Conference on Task Models and Diagrams*, pp. 99–104. ACM Press, New York, 2004.

[Her00] T. Herrmann. Evolving Workflows by User-driven Coordination. In R. Reichwald; J. Schlichter (Eds.) (2000), Tagungsband D-CSCW 2000, pp. 103-114. Teubner, 2000.

[HHK04] M. Hammori, J. Herbst, and N. Kleiner. Interactive Workflow Mining. In: J. Desel, B. Pernici, and M. Weske (Eds.): *BPM 2004, LNCS 3080*, pp. 211–226. Springer-Verlag Berlin Heidelberg, 2004.

[HMBR05] H. Holz, H. Maus, A. Bernardi, and O. Rostanin. From Lightweight, Proactive Information Delivery to Business Process-Oriented Knowledge Management. *Journal of Universal Knowledge Management*, vol. 0, no. 2 (Nov. 2005), pp. 101-127, 2005.

[Hol95] D. Hollingsworth. *The Workflow Reference Model*. Workflow Management Coalition, Document Number TC00-103, Winchester, Hampshire, UK, 1995.

[HRD+06] H. Holz, O. Rostanin, A. Dengel, T. Suzuki, K. Maeda, and K. Kanasaki. Task-based process know-how reuse and proactive information delivery in TaskNavigator. In: *Proceedings of the 15th ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 522–531. ACM Press, New York, 2006.

[IMB+03] IBM Inc., Microsoft Corp., BEA Inc., SAP AG, Siebel Systems Inc. *The Business Process Execution Language for Web Services, version 1.1*. URL http://www.ibm.com/developerworks/library/specification/ws-bpel/ [07.11.2008].

[JBoss] JBoss, URL http://jboss.org/ [14.01.2009].

[jBPM] JBoss Business Process Management, URL http://www.jboss.org/jbossjbpm/ [14.01.2009].

[JGOR07] H. M. Jarodzka, O. Grebner, E. Ong, and U. V. Riss. Motivate users to construct collective knowledge via IT. In: *WM'07, 4th Conference for Professional Knowledge Management*, Potsdam, Germany, pp. 372-381. GITO Verlag, Berlin, 2007.

[JK06a] B. E. John and D. E. Kieras. The GOMS family of analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction*, vol. 3 no. 4 (December, 1996), pp. 320–351. ACM Press, New York, 1996.

[JK06b] B. E. John and D. E. Kieras. Using GOMS for User Interface Design and Evaluation: Which Technique? *ACM Transactions on Computer-Human Interaction*, vol. 3 no. 4 (December, 1996), pp. 287–319. ACM Press, New York, 1996.

[JOA03] M. E. Jennex, L. Olfman, and T. B. A. Addo. The Need for an Organizational Knowledge Management Strategy. In: *Proceedings of the 36th annual Hawaii International Conference on System Sciences*, p. 117.1. IEEE Computer Society, Washington, 2003.

[Joh03] B. E. John. Information processing and skilled behavior. In J.M. Carroll (Ed.): *HCI Models, Theories and Frameworks: Toward a multidisciplinary science*, pp. 55-101, Morgan Kaufmann, San Francisco, 2003.

[Jor04] H. D. Jorgensen. *Interactive Process Models*. Ph.D. Thesis, Norwegian University of Science and Technology, Trondheim, Norway, 2004.

[Kie04] D. Kieras. GOMS Models for Task Analysis. In: Dan Diaper and Neville Stanton (Eds.): *The Handbook of Task Analysis for Human-Computer Interaction*, pp. 83-116. Lawrence Erlbaum Associates, 2004.

[KK05] T. Klug and J. Kangasharju. Executable Task Models. In: *Proceedings of the 4th international workshop on Task models and diagrams*, pp. 119-122. ACM Press, New York, 2005.

[Kle04] N. Kleiner. Supporting Usage-Centered Workflow Design: Why and How? In: J. Desel, B. Pernici, and M. Weske (Eds.): *BPM 2004, LNCS 3080*, pp. 227–243. Springer-Verlag Berlin Heidelberg, 2004.

[KM04] A. J. Ko and B. A. Myers. Designing the Whyline: A Debugging Interface for Asking Questions about Program Behavior. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pp. 151-158. ACM Press, New York, 2004.

[KNS92] G. Keller, M. Nüttgens, and A.-W. Scheer. Semantische Prozeßmodellierung auf der Grundlage „Ereignisgesteuerter Prozeßketten (EPK)." In: A.-W. Scheer (Ed.): *Veröffentlichungen des Instituts für Wirtschaftsinformatik*, no. 89, Saarbrücken: Universität des Saarlands, 1992. URL http://www.iwi.uni-sb.de/Download/iwihefte/heft89.pdf [06.11.2008].

[Kun03] M. Kuniavsky. *Observing the User Experience: A Practitioner's Guide to User Research*. Morgan Kaufmann Publishers, 2003.

[LB03] C. Larman and V. Basili. Iterative and incremental developments, a brief history. *Computer*, vol. 36, no. 6 (June 2003), pp. 47-56. IEEE Computer Society, 2003.

[Let06] C. Letondal. Participatory Programming: Devleloping Programmable Bioninformatics Tools for End-Users. In: Henry Lieberman, Fabio Paternó and Volker Wulf (Eds.): *End-User Development*, pp. 207-242. Springer, 2006.

[Lie01] H. Lieberman. *Your Wish is My Command: Programming by Example*. Morgan Kaufmann, 2001.

[Lik32] R. Likert. A technique for the Measurement of Attitudes. *Archives of Psychology*, no. 140, pp. 1-50.

[LPKW06] H. Lieberman, F. Paternó, M. Klann, and V. Wulf. End-User Development: An Emerging Paradigm. In: Henry Lieberman, Fabio Paternó and Volker Wulf (Eds.): *End-User Development*, pp. 1-8. Springer, 2006.

[LV04] Q. Limbourg and J. Vanderdonckt. Comparing Task Models for User Interface Design. In: Dan Diaper and Neville Stanton (Eds.): *The Handbook of Task Analysis for Human-Computer Interaction*, pp. 135 -154. Lawrence Erlbaum Associates, 2004.

[Mac90] W. Mackay. Patterns of sharing customizable software. In: *Proceedings of the 1990 ACM conference on Computer-supported cooperative work (CSCW)*, pp. 209-221. ACM Press, New York, 1990.

[MCLM90] A. MacLean, K. Carter, L. Lövstrand, and T. Moran. User-tailorable systems: pressing the issues with buttons. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pp. 175-182. ACM Press, New York, 1990.

[MG99] Y. Malhotra and D. F. Galletta. Extending the Technology Acceptance Model to Account for Social Influence: Theoretical Bases and Empirical Validation. In: *Proceedings of the 32nd Annual Hawaii International Conference on Proceedings of the Hawaii International Conference on System Sciences (HICSS 32)*, pp. 14. IEEE Computer Society, Washington, 1999.

[Min08] Mindjet. URL http://www.mindjet.com [16.10.2008].

[MM00] A. Mørch and N. Mehandjiev. Tailoring as Collaboration: The Mediating Role of Multiple Representations and Application Units. *Computer Supported Cooperative Work*, vol. 9, no 1 (2000), pp. 75-100. Kluwer Academic Publishers, 2000.

[MPS02] G. Mori, F. Paternó, and C. Santoro. CTTE: Support for Developing and Analyzing Task Models for Interactive System Design. In: *IEEE Transactions on Software Engineering*, vol. 28, no. 9 (September 2002). IEEE Computer Society, 2002.

[MSG+08] N. Mehanjiev, T. Stoitsev, O. Grebner, S. Scheidl, and U. Riss. End-User Development for Task Management: Survey of Attitudes and Practices. In: *Proceedings 2008 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Herrsching am Ammersee, Germany, pp. 166-174. IEEE Computer Society, 2008.

[MSL06] N. Mehandjiev, A. Sutcliffe, and D. Lee. Organizational View of End-User Development. In: Henry Lieberman, Fabio Paternó and Volker Wulf (Eds.): *End-User Development*, pp. 371-399. Springer, 2006.

[Mül05] J. Müller. *Workflow-based Integration: Grundlagen, Technologien, Management*. Springer-Verlag Berlin Heidelberg, 2005.

[MW47] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, vol. 18, pp. 50-60. Institute of Mathematical Statistics, 1947.

[Nar93] B. Nardi. *A Small Matter of Programming: Perspectives on End User Computing*. MIT Press, 1993.

[Nat08] National Instruments, 2008. LabVIEW. URL http://www.ni.com/labview/ [24.10.2008].

[NEP06] NEPOMUK Project, 2006. *The Social Semantic Desktop*. URL http://nepomuk.semanticdesktop.org [14.01.2009].

[NM90] B. Nardi and J. Miller. An Ethnographic Study of Distributed Problem Solving in Spreadsheet Development. In: *Proceedings of the 1990 ACM conference on Computer-Supported Cooperative Work (CSCW)*, pp. 197 – 208. ACM Press, New York, 1990.

[OG94] W. J. Orlikowski and D. C. Gash. Technological Frames: Making Sense of Information Technology in Organizations. *ACM Transactions of Information Systems*, vol. 12, no. 2 (April 1994), pp. 174 – 207, ACM Press, New York, 1994.

[OGR07] E. Ong, O. Grebner, and U. V. Riss. Pattern-based Task Management: Pattern Lifecycle and Knowledge Management. In: *WM'07, 4th Conference for Professional Knowledge Management*, Potsdam, Germany, pp. 357-364. GITO Verlag, Berlin, 2007.

[OMG06] Object Management Group. *Business Process Modeling Notation (BPMN) Specification*. OMG document dtc/06-02-01, URL http://www.bpmn.org/Documents/ [15.01.2009]

[Ous98] J. Ousterhout. Scripting: High Level Programming for the 21st Century. In: *Computer*, vol. 31, no. 3 (March 1998), pp. 23 – 30. IEEE Computer Society, 1998.

[Pat00] F. Paternó. *Model-Based Design and Evaluation of Interactive Applications*. Springer, Heidelberg, 2000.

[Pat04] F. Paternó. ConcurTaskTrees: An Engineered Notation for Task Models. In: Dan Diaper and Neville Stanton (Eds.): *The Handbook of Task Analysis for Human-Computer Interaction*, pp. 483-501. Lawrence Erlbaum Associates, 2004.

[PB04] P. Palanque and S. Basnyat. Task Patterns for Taking into Account in an Efficient and Systematic Way Both Standard and Abnormal User Behaviour. In: *Proceedings of IFIP 13.5 Working Conference on Human Error, Safety and Systems Development*, Toulouse, France, pp. 109–130, 2004.

[PBC06] M. Prabaker, L. Bergman, and V. Castelli. An Evaluation of Using Programming by Demonstration and Guided Walkthrough Techniques for Authoring and Utilizing Documentation. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pp. 241-250. ACM Press, New York, 2006.

[PJAA96] S. Page, T. Johnsgard, U. Albert, and C. Allen. User Customization of a Word Processor. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pp. 340 – 346. ACM Press, New York, 1996.

[PM06] J. Pane and B. Myers. More Natural Programming Languages and Environments. In: Henry Lieberman, Fabio Paternó and Volker Wulf (Eds.): *End-User Development*, pp. 31-50. Springer, 2006.

[PMM97] F. Paternó, S. Mancini, and S. Meniconi. ConcurTaskTree: a diagrammatic notation for specifying Task Models. In: *Proceedings of Interact 1997*, pp. 362–369. Chapman & Hall, Boca Raton, 1997.

[PW08] L. Priese and H. Wimmel. *Petri-Netze*. Springer-Verlag Berlin Heidelberg, 2008.

[RC07] T. Rattenbury and J. Canny. CAAD: An Automatic Task Support System. In: *Proceedings of the 2007 SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pp. 687-696. ACM Press, New York, 2007.

[RCKM06] U. V. Riss, U. Cress, J. Kimmerle, and S. Martin. Knowledge Transfer by Sharing Task Patterns - From Experiment to Application. In: *J. S. Edwards (Ed.), Proc. of KMAC 2006, The Third Knowledge Management Aston Conference*, pp. 121-133. Operational Research Society, Birmingham, 2006.

[RD98] M. Reichert and P. Dadam. ADEPTflex – supporting dynamic changes of workflows without losing control. *Journal of Intelligent Information Systems (JIIS)*, Special Issue on Workflow Management Systems, vol. 10, no. 2 (March/April 1998), pp. 93-129, 1998.

[Rei05] H. A. Reijers. Process Design and Redesign. In: Marlon Dumas, Wil van der Aalst, Arthur ter Hofstede (Eds.): *Process-Aware Information Systems. Bridging People and Software through Process Technology*, pp. 363-395. John Wiley & Sons, 2005.

[RI06] A. Repenning and A. Ioannidou. What Makes End-User Development Tick? 13 Design Guidelines. In: Henry Lieberman, Fabio Paternó and Volker Wulf (Eds.): *End-User Development*, pp. 51-85. Springer, 2006.

[RIG07] J. Recker, M. Indulska, and P. Green. Extending Representational Analysis: BPMN User and Developer Perspectives. In: G. Alonso, P. Dadam, and M. Rosemann (Eds.): *BPM 2007, LNCS 4714*, pp. 384–399. Springer-Verlag Berlin Heidelberg, 2007.

[RJS02] A. Ribak, M. Jacovi, and V. Soroka. "Ask Before You Search" Peer Support and Community Building with ReachOut. In: *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pp. 126–135. ACM Press, New York, 2002.

[RM08] H. A. Reijers and A. Mendling. Modularity in Process Models: Review and Effects. In: M. Dumas, M. Reichert, and M.-C. Shan (Eds.): *BPM 2008, LNCS 5240*, pp. 20–35. Springer-Verlag Berlin Heidelberg. 2008.

[RRD03] S. Rinderle, M. Reichert, and P. Dadam. Evaluation of Correctness Criteria for Dynamic Workflow Changes. In: W.M.P. van der Aalst et al. (Eds.): *BPM 2003, LNCS 2678*, pp. 41–57. Springer-Verlag Berlin Heidelberg, 2003.

[RRMvdA05] U. Riss, A. Rickayzen, H. Maus, and W. M. P. van der Aalst. Challenges for Business Process and Task Managemen. *Journal of Universal Knowledge Management*, vol. 0, no. 2 (2005), pp. 77-100.

[SAMS01] S. Schwarz, A. Abecker, H. Maus, and M. Sintek. Anforderungen an die Workflow-Unterstützung für wissensintensive Geschäftsprozesse. In: *WM'01, 1st Conference for Professional Knowledge Management*, Baden-Baden, Germany, 2001.

[Sch03] S. Schwarz. Task-Konzepte: Struktur und Semantik für Workflows. In: *WM'03, 2nd Conference for Professional Knowledge Management, GI LNI P-28*, pp. 351-356. Bonner Köllen Verlag, 2003.

[Sch08] C. Schneider. *Graphische Integration bestehender Informationen zur kontextualisierten Unterstützung wissensintensiver Arbeitsplätze*. Degree dissertation, Technical University Darmstadt, 2008.

[Sch09] B. Schmidt. *Task Pattern Generation and Use on the Semantic Desktop. Applied Experience Management*. Master Thesis, University Paderborn, Germany, 2009.

[SCT01] D. C. Smith, A. Cypher, and L. Tesler. Novice Programming Comes of Age. In: *Your Wish Is My Command: Programming By Example*, Morgan Kaufmann, 2001.

[SDW08] M. Spahn, C. Dörner, and V. Wulf. End User Development: Approaches Towards a Flexible Software Design. In: *Proceedings of 16th European Conference on Information Systems National University of Ireland, Galway*, June 9-11, 2008.

[Sin04] D. Sinnig. *The Complicity of Patterns and Model-based UI Development*. Master's Thesis in Department of Computer Science, Concordia University, Montreal, Canada, 2004.

[SIT06] N. Siu, L. Iverson, and A. Tang. Going with the Flow: Email Awareness and Task Management. In: *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pp. 441–450. ACM Press, New York, 2006.

[SLM03] A. Sutcliffe, D. Lee, and N. Mehandjiev. Contributions, Costs and Prospects for End-User Development. In: *Proceedings of the Tenth International Conference on Human-Computer Interaction*, vol. 3. Lawrence Erlbaum Associates, 2003.

[SPSS08] SPSS Inc. URL http://www.spss.com/ [29.02.2009]

[SQK06] G. Stevens, G. Quaisser, and M. Klann. Breaking It Up: An Industrial Case Study of Component-Based Tailorable Software Design. In: Henry Lieberman, Fabio Paternó and Volker Wulf (Eds.): *End-User Development*, pp. 269-294. Springer, 2006.

[SS08a] T. Stoitsev and S. Scheidl. An Architecture for End-User Driven Business Process Management. In: *Proceedings of the 2008 IEEE Enterprise Distributed Object Computing Conference (EDOC)*, pp. 157-165. IEEE Computer Society, 2008.

[SS08b] T. Stoitsev and S. Scheidl. A Method for Modeling Interactions on Task Representations in Business Task Management Systems. P. Forbrig and F. Paternò (Eds.): *HCSE/TAMODIA 2008, LNCS 5247*, pp. 84–97. Springer, 2008.

[SSFM08a] T. Stoitsev, S. Scheidl, F. Flentge, and M. Mühlhäuser. Enabling End Users to Proactively Tailor Underspecified, Human-Centric Business Processes: "Programming by Example" of Weakly-Structured Process Models. In: J. Filipe and J. Cordeiro (Eds.): *ICEIS 2008, LNBIP 19*, pp. 307–320, Springer, 2009.

[SSFM08b] T. Stoitsev, S. Scheidl, F. Flentge, and M. Mühlhäuser. Architecture for End-User Driven Composition of Underspecified, Human-Centric Business Processes. In: *Proceedings of the 10th International Conference on Enterprise Information Systems (ICEIS)*, Barcelona, Spain, pp. 165-172, 2008.

[SSFM08c] T. Stoitsev, S. Scheidl, F. Flentge, and M. Mühlhäuser. Enabling End-User Driven Business Process Composition through Programming by Example in a Collaborative Task Management System. In: *Proceedings of the 2008 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Herrsching am Ammersee, Germany, pp. 53-62. IEEE Computer Society, 2008.

[SSFM08d] T. Stoitsev, S. Scheidl, F. Flentge, and M. Mühlhäuser. From Personal Task Management to End-User Driven Business Process Modeling. In: M. Dumas, M. Reichert, and M.-C. Shan (Eds.): *BPM 2008, LNCS 5240*, pp. 84–99. Springer-Verlag Berlin Heidelberg, 2008.

[SSS07] T. Stoitsev, S. Scheidl, and M. Spahn. A Framework for Light-Weight Composition and Management of Ad-Hoc Business Processes. In: Winckler, M., Johnson, H., and Palanque, P. (Eds.): *TAMODIA 2007, LNCS 4849*, pp. 213-226. Springer-Verlag Berlin Heidelberg, 2007.

[STA05] A. W. Scheer, O. Thomas, and O. Adam. Process Modeling Using Event-Driven Process Chains. In: Marlon Dumas, Wil van der Aalst, Arthur ter Hofstede (Eds.): *Process-Aware Information Systems. Bridging People and Software through Process Technology*, pp. 119-145. John Wiley & Sons, 2005.

[Suc87] L. Suchman. Plans and situated actions: The problem of human-machine communication. Cambridge University Press, New York, 1987.

[Sut05] A. Sutcliffe. Evaluating the Costs and Benefits of End-User Development. In: *ACM SIGSOFT Software Engineering Notes*, vol. 30, no. 4 (July 2005), Session: Workshop on End-User Software Engineering (WEUSE), pp. 1-4. ACM Press, New York, 2005.

[SW65] S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, vol. 52, no. ¾ (December 1965), pp. 591–611.

[Swe93] K. D. Swenson. A Visual Language to Describe Collaborative Work. In: *Proceedings of the IEEE Symposium on Visual Languages*, pp. 298-303. IEEE Computer Society, 1993.

[vdA99] W. M. P. van der Aalst. Formalization and Verification of Event-driven Process Chains. *Information and Software Technology*, vol. 41, no. 10 (1999), pp. 639-650. Elsevier Science B.V., 1999.

[vdABV+99] W. M. P. van der Aalst, T. Basten, H. M. W. Verbeek, P. A. C. Verkoulen, and M. Verhoeve. Adaptive Workflow: On the interplay between flexibility and support. In: *Proceedings of the first International Conference on Enterprise Information Systems (ICEIS)*, Setubal, Portugal, pp. 353–360, 1999.

[vdAHW03] W. M. P. van der Aalst, A. Hofstede, and M. Weske. Business Process Management: A Survey. In: W. M. P. van der Aalst et al. (Eds.): *BPM 2003, LNCS 2678*, pp. 1–12. Springer-Verlag Berlin Heidelberg, 2003.

[vdAS04] W. M. P. van der Aalst and M. Song: Mining Social Networks: Uncovering interaction patterns in Business Processes. In: J. Desel, B. Pernici, and M. Weske (Eds.): *BPM 2004, LNCS 3080*, pp. 244–260. Springer-Verlag Berlin Heidelberg, 2004.

[vdAvH02] W. M. P. van der Aalst and K. van Hee. *Workflow Management. Models, Methods, and Systems*. MIT Press, 2002.

[vdAW03] W. M. P. van der Aalst and A. Weijters: Process mining: a research agenda. *Computers in Industry*, vol. 53 (2003), Elsevier B.V., 2003.

[vdAW05] W. M. P. van der Aalst and A. Weijters. Process Mining. In: Marlon Dumas, Wil van der Aalst, Arthur ter Hofstede (Eds.): *Process-Aware Information Systems. Bridging People and Software through Process Technology*, pp. 235-255. John Wiley & Sons, 2005.

[vdAWG05] W. M. P. van der Aalst, M. Weske, and D. Grünbauer. Case Handling: A New Paradigm for Business Process Support. *Data and Knowledge Engineering*, vol. 53, no. 2 (May 2005), pp. 129–162. Elsevier B.V., 2005.

[vdVLB96] G. van der Veer, B. Lenting, and B. Bergevoet. GTA: Groupware task analysis - modeling complexity. *Acta Psychologica*, vol. 91, pp. 297–322, 1996.

[vdVvW04] G. van der Veer and M. van Weile. DUTCH: Designing for Users and Tasks from Concepts to Handles. In: Dan Diaper and Neville Stanton (Eds.): *The Handbook of Task Analysis for Human-Computer Interaction*, pp. 155 -173. Lawrence Erlbaum Associates, 2004.

[Ver04] L. Verner. BPM: The Promise and the Challenge. *ACM Queue* vol. 2, no. 1 (March 2004), pp. 82-91. ACM Press, 2004.

[VVK08] J. Vanhatalo, H. Völzer, and J. Koehler. The Refined Process Structure Tree. In: M. Dumas, M. Reichert, and M.-C. Shan (Eds.): *BPM 2008, LNCS 5240*, pp. 100–115. Springer-Verlag Berlin Heidelberg, 2008.

[Wes07] M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, 2007.

[Wii04] K. M. Wiig. *People-focused knowledge management: how effective decision making leads to corporate success*. Elsevier Butterworth–Heinemann, 2004.

[Win86] T. Winograd. A language/action perspective on the design of cooperative work. In: *Proceedings of the 1986 ACM Conference on Computer-Supported Cooperative Work*, pp. 203-220. ACM Press, New York, 1986.

[WJ04] V. Wulf and M. Jarke. The Economics of End-User Development. *Communications of the ACM*, vol. 47, no. 9 (Sep. 2004). ACM Press, 2004.

[WR95] V. Wulf and M. Rohde. Towards an Integrated Organization and Technology Development. In: *Proceedings of the Symposium on Designing Interactive Systems*, pp. 55-64. ACM Press, New York, 1995.

[WSW06] M. Won, O. Stiemerling, and V. Wulf. Component-Based Approaches to Tailorable Systems. In: Henry Lieberman, Fabio Paternó and Volker Wulf (Eds.): *End-User Development*, pp. 115-141. Springer, 2006.

[WvdAD+06] P. Wohed, W. M. P. van der Aalst, M. Dumas, A. H. M. ter Hofstede, and N. Russell. On the Suitability of BPMN for Business Process Modelling. In: S. Dustdar, J.L. Fiadeiro, and A. Sheth (Eds.): *BPM 2006, LNCS 4102*, pp. 161–176. Springer-Verlag Berlin Heidelberg, 2006.

[W3C00] World Wide Web Consortium (W3C). *Simple Object Access Protocol (SOAP)*. Version 1.1. URL http://www.w3.org/TR/soap/

[W3C02] World Wide Web Consortium (W3C). *XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition)*. URL http://www.w3.org/TR/xhtml1/

[W3C03] World Wide Web Consortium (W3C). *Extensible Markup Language (XML)*. URL http://www.w3.org/XML/

[W3C05] World Wide Web Consortium (W3C). *SOAP Message Transmission Optimization Mechanism*. URL http://www.w3.org/TR/soap12-mtom/

Appendix A: Interview Guidelines for the Preliminary Empirical Studies

This appendix contains the interview guidelines in the way that these were used in the preliminary empirical studies. An important notice here is that they are all translated from German, which was the original language in which the interviews were held. The first part contains the guidelines for the start interviews, which were performed during the first phase of the field studies. The second part contains the interviews, used for the elaboration on concrete processes and the creation of Hierarchical Task Analysis (HTA) for describing the related tasks.

A.1 Interview Guideline for Start Interviews

Target group
Target group for the interviews is company management, respectively managing employees of the financial, planning and IT departments of the Small and Medium Enterprises (SMEs).

Introduction:

- *Duration*: The planned duration of the interviews is about 90 minutes.
- *Privacy*: The interview contents are confidential and will be not be made accessible outside the organizations from the EUDISMES project. Within these companies, the contents will be accessible only to employees, involved in the project. Interview data will be published only in anonymous and aggregated form. The personal data of the interviewees will be stored separately from the interview contents.
- *Recording*: The interviews will be recorded through an audio recorder and additionally notes will be made.
- *Consent to record*: Obtaining the interviewee consent to record the interview is necessary. If this is not granted, the recording will be stopped immediately.
- *Legal*: If necessary, the elicitation and use of the empirical data will be discussed with the works committee or personnel council.

Personal data of interviewee

- What is your Name?
- Could you tell us your age?
- How long have you been employed at the company?
- What is your position at the company?
- Could you give a short description of your personal experience in the area of Information Technology (IT) and company planning?

1) Basic data

- How old is the company?
- What is the business branch, i.e. typical assignments or products?
- What is the work load/performance?
- What is the number of employees?
- What is the turnover?

2) Software

- What software applications do you use in your company?

    - What is the IT architecture in general?
    - Are SAP modules used, and if yes, which modules?
    - What office applications are used?
    - Is further software used, e.g. for project management, planning, workflow, task management, monitoring, reporting or other like CAD/CAM and further domain-specific software?
- What systems and networks do you use in your company? Is there further hardware that is connected to the software infrastructure like e.g. machines, production lanes, scanner, printer etc.
- Do you use internet-based communication or cooperation systems like e.g. web-email?

3) Software support

- Who is using the described technology – complete enterprise or specific departments? What roles are there for this usage (key-user etc.)?
- Do you work also with external partners like e.g. IT consultants or developers? How does this cooperation work?
- What is the specialization/area of expertise the departments?
- Are there technical problems in the interfaces between the systems and how do you manage such problems?

4) Organization: Company planning and management

- How is the project and department planning performed?
- How does the capacity and resource planning take place?
- How does the financial planning take place? What is specified, how from whom?
- How are responsibilities assigned? Is there a defined hierarchy for the assignments?
- Do you distribute parts of a project to external partners (outsourcing)?
- How does the order management function?

5) Organization: Work distribution and cooperation in the software management

- What are the structure and the work distribution in the software management?
- What departments in your company are responsible for software management? How do you support the users in the other departments?
- Who is responsible for the software planning: requirements analysis, acquisition, adaptation and education of the staff?
- Do you have your own software development? If yes, for which departments? For which tasks is software development needed and how do the corresponding processes look like?
- Do you have networks where you can obtain support on problems?
    - Contact person by technical questions?
    - Contact person by configuration problems?
    - Do some external partners take over some tasks of the software management (outsourcing)?

6) Special questions related to task management (optional):

- *Process scenarios*: What business or work processes are of interest for you and why?
    - Standardization/Auditing of existing processes
    - Optimization of existing processes through integration and design
    - Transparency in hidden processes, which exist but are not explicated and not formally supported
    - Creation of new processes
    - Processes involving other partner companies

    - Processes, involving multiple systems
- *Process structures*: Can you describe some core processes, which:
    - to a certain extent have a defined structure
    - reappear in the same or similar manner in day-to-day activities
    - require a high degree of communication and consolidation between the employees
    - in which the employees can be more efficient if they can inspect similar cases in an archive
    - require insight and understanding of the overall process beyond the individual contribution in this process through their personal tasks
    - are not supported through possibly existing current workflow systems
- *Cooperation between employees*: What form of communication is used or would be ideally used if available?
    - Workflow systems
    - Email
    - Telephone
    - Personal face-to-face talk
    - Web-based communication over internet (from field, home etc.)
- *Insight*: Which employees with openness and some vision can we interview, i.e. as key- or end users, to elaborate on their problems, wishes and visions for the work with workflow and collaboration systems?

7) Special questions related to analytics and reporting (optional)

- How important is it in your company to be able to create ad-hoc reports and data analysis?
- How many employees require such reports and how many are able to create these?
- What is the potential number of users?
- How important is thereby the interoperability between different systems?
- How important is thereby the integration with Microsoft Excel?
- What are the current problems in the area of analytics and reporting?

8) General

- How would you describe your most important questions with the management of the software infrastructure? Is something in particular to mention, that did not come up so far?
- Where do you see possibly the need to make things better?
- Could you direct us to end users for further interviews and evaluation?

A.2 Interview Guideline for Process-Focused Interviews

Target group
Target group for the interviews comprises employees from different company departments, such as IT, sales, purchase, marketing etc., who participate in processes that involve a high degree of cooperation and coordination. The processes are informal and are not supported through business process management or workflow management systems.

Introduction:

- *Duration*: The planned duration of the interviews is about 90 minutes.
- *Privacy*: The interview contents are confidential and will be not be made accessible outside the organizations from the EUDISMES project. Within these companies, the contents will be accessible only to employees, involved in the project. Interview data will be published only in anonymous and aggregated form. The personal data of the interviewees will be stored separately from the interview contents.
- *Recording*: The interviews will be recorded through an audio recorder and additionally notes will be made.
- *Consent to record*: Obtaining the interviewee consent to record the interview is necessary. If this is not granted, the recording will be stopped immediately.
- *Legal*: If necessary, the elicitation and use of the empirical data will be discussed with the works committee or personnel council.

Personal data of interviewee
These questions are relevant only for Interviewees who have not participated in the previous interview phases, i.e. for which this information is not already available.

- What is your Name?
- Could you tell us your age?
- How long have you been employed at the company?
- What is your position at the company?
- Could you give a short description of your personal experience in the area of Information Technology (IT) and company planning?

1) Task flow/task structure
The goal of the questions from this section is to get an overview of which tasks (and sub-tasks) are part of an informal business process. During the interview, the tasks should be documented in a way that is clear also to the interviewee, for example through post-its. The interviewee can then confirm that the interviewer has understood things correctly. Using the structure of a Hierarchical Task Analysis (HTA) is a possible way to go.

After the task flow/task structure is identified, a second walkthrough should be performed, where questions about artifacts or collaboration on the various tasks and sub-tasks are asked.

If interviews are made with several people working on the same process, one HTA can be constructed for each individual, or one larger HTA structure can be build up, using the input of all interviewees. The latter approach is preferable as one final, consolidated HTA should be constructed in the end from the interview results, showing the whole process or at least the part of it for which the interviewees are responsible.

The questions in this section are asked iteratively until the process is identified down to the lowest relevant level of subtasks.

- When you start your work on subject X/task Y/process Z, what do you do first?
- How do you do this? (sub-tasks)
- What do you do next? How?
- What is the trigger for this task?

2) Artifacts

For this section it is important to assure that all artifacts in relation to the respective task(s) are documented.

- What information do you need in order to be able to do this task?
- From whom/from where/how do you get this information?
- Do you have to do something to get the information (phone call, search of files etc), or do you get it automatically (sent via email etc.)?
- When you finish this task, what is the output/result?
- Is the output documented (in document, spread sheet, form etc.) or/and transferred (email, phone call, face to face communication)?

3) Dependencies

- Are your tasks sometimes dependent on each other? How (time, in-output)?

4) Collaboration

- Are you dependent on colleagues who perform some task in your process?
- In which tasks are colleagues involved? How? What do they do?
- Do you sometimes ask a colleague to do a task/part of a task for you? How do you ask (email, telephone)?
- When you ask a colleague to do a task, is it important that you get confirmation that she/he can or is willing to perform the task? Why (not)?
- If a colleague has accepted a task, is it important for you to know when she/he has finished it or how much of it she/he has achieved so far?
- *(For project leaders)* When a task has been performed by a colleague, is the performance quality (how well the task was performed) of importance? Do you make notice of this? Why (not)? Would you like to have more possibilities to document performance?

5) Current problems

- What problems exist in your work on this process (missing info, communication problems, redundant tasks etc.)?

---------- **Questions independent from task structure** ----------

6) Patterns – lifecycle

- Do you sometimes make notes about how to perform a task or where to find certain information (as memory support for yourself)?
- How often do you consult these notes?
- Do you sometimes change/optimize these notes (as you learn new things or discover a better way to perform a task)?
- Do you sometimes ask a colleague, who may have more experience in a certain area, for advice on how to perform a task?
- Do colleagues sometimes ask you for advice on how to perform a task?
- Would it be helpful to you if there were some sort of guidance (instructions, step-by-step etc.) as to how to perform a task that is new to you?

- If your activities during the performance of a task were automatically documented, would this be helpful to you?
- Would it be helpful to you the next time you performed a task, if you could manually optimize the automatically generated version of the first run-through of the task?

7) Communication

- How do you prefer to communicate with your colleagues (face to face, email, phone, other)?
- If more central documentation about tasks would eliminate the need to talk (face-to-face, phone) to your colleagues, but instead communicate over electronic way such as for example email, how would you feel about that?

8) Transparency

- Do you always know how your tasks fit into the 'big picture'? Do you know why you perform all tasks/why a certain task is needed?
- Would you like to know more about 'the big picture', i.e. how your own task fit in, what other tasks are part of the process, who is performing the other tasks, what status they have etc. Why would this (not) be helpful?
- Would this information make you more efficient in your own work? How?
- Would it be OK if colleagues could see what tasks you are working on?

Appendix B: Hierarchical Task Analysis Diagrams

This appendix contains Hierarchical Task Analysis (HTA) [AS00, Ann04] diagrams, resulting from the third phase of the empirical field studies described in Chapter 2. Including all elaborated HTA diagrams from the second phase would burst the scope of the dissertation. Therefore only an **HTA for a process for initiation of consignation sales** is provided in the following.

An overview of the generic tasks is provided first, followed by detailed HTA diagrams for the tasks of different departments. Unless explicitly stated through a corresponding legend in the top right corner of a given task, all tasks given in the following figures belong to the responsibility area of the department stated in the figure title.

The lifecycle of the different artifacts (documents) used throughout the process are provided finally, to describe the document flow.

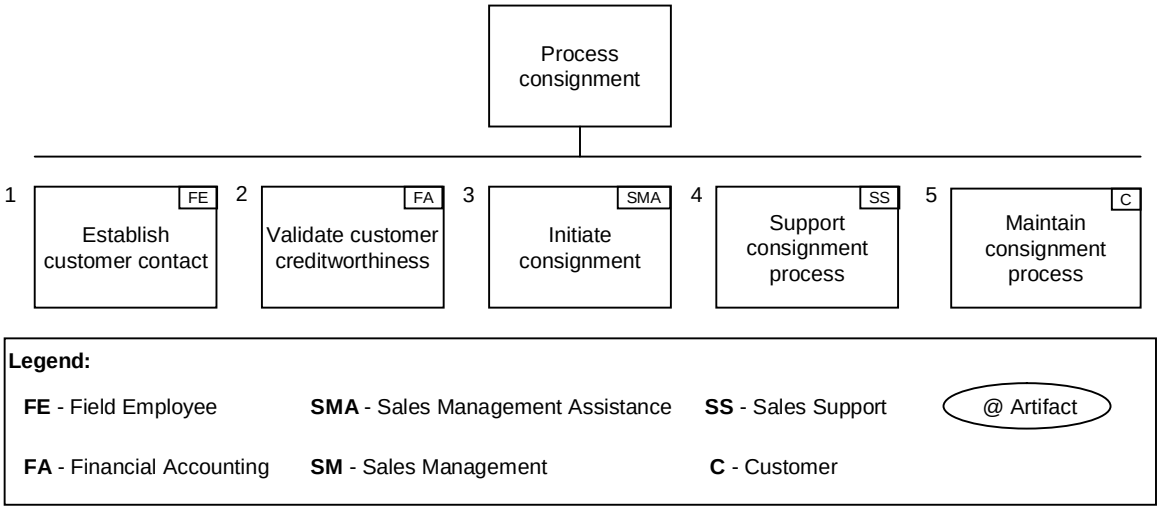


Figure B-1: Overview of process for initiation of consignation sales

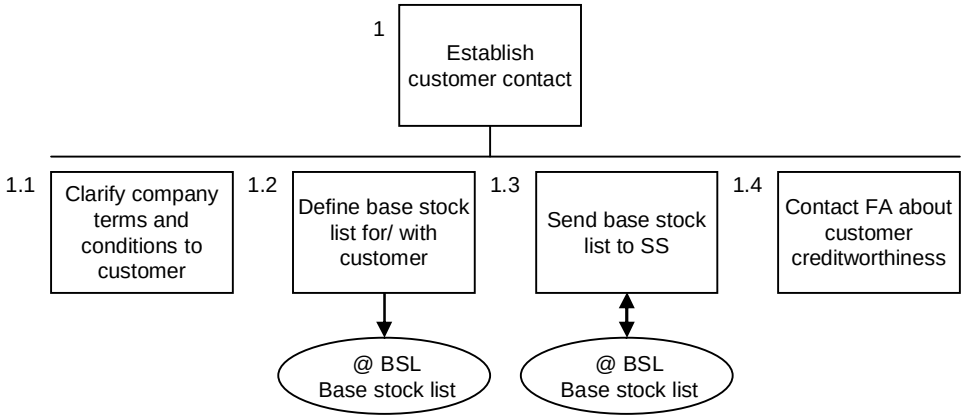


Figure B-2: Tasks of Field Employee (FE)

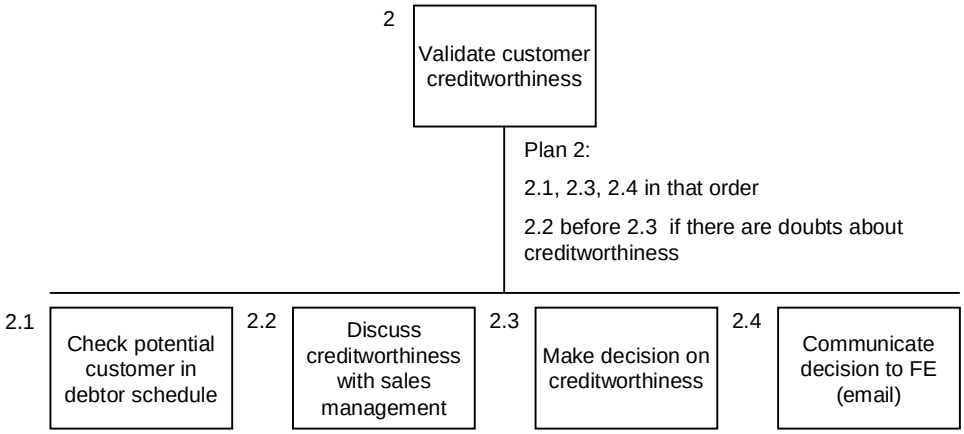


Figure B-3: Tasks of Financial Accounting (FA)

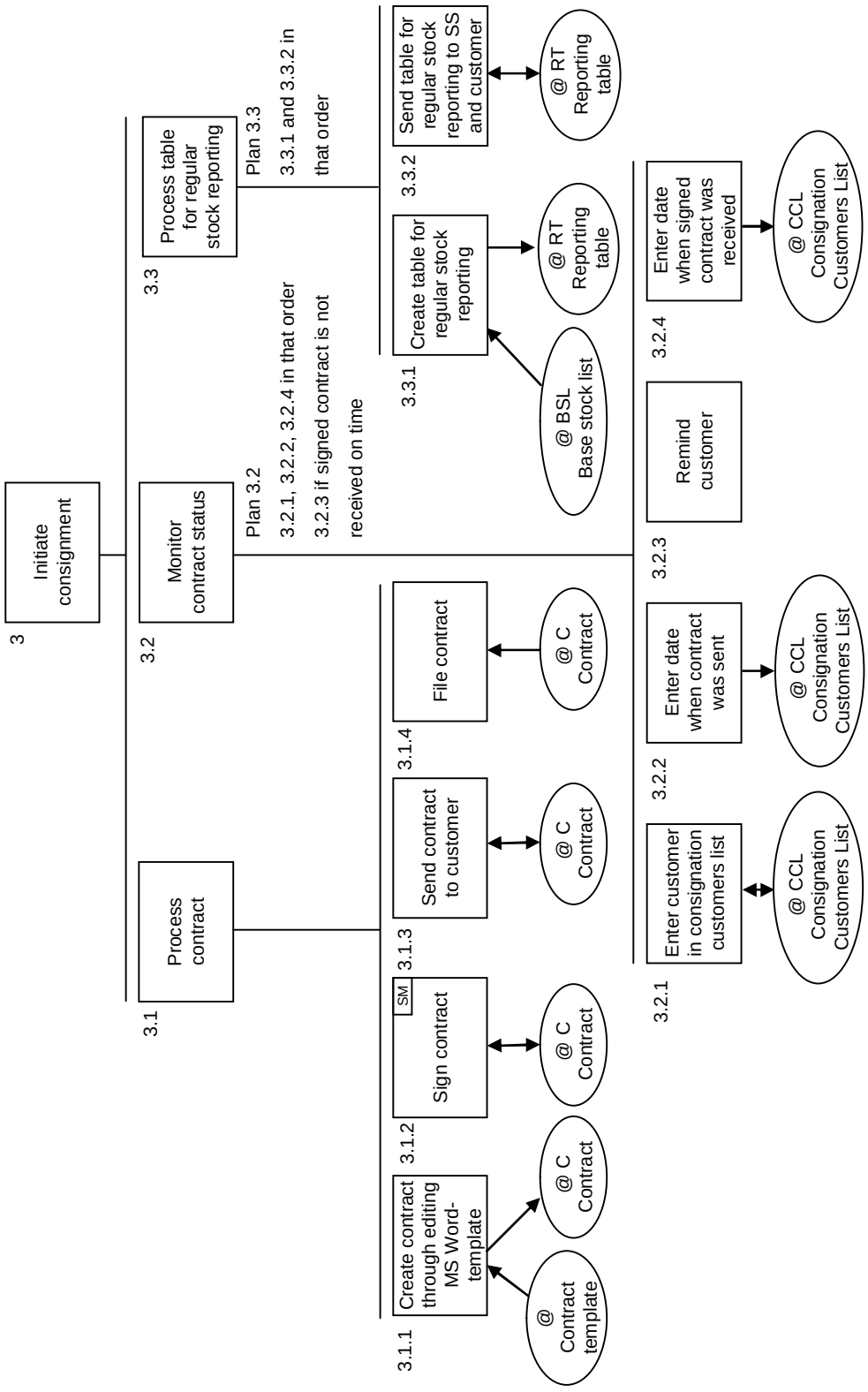


Figure B-4: Tasks of Sales Management (Assistance) (SM(A))

Figure B-5: Tasks of Sales Support (SS)

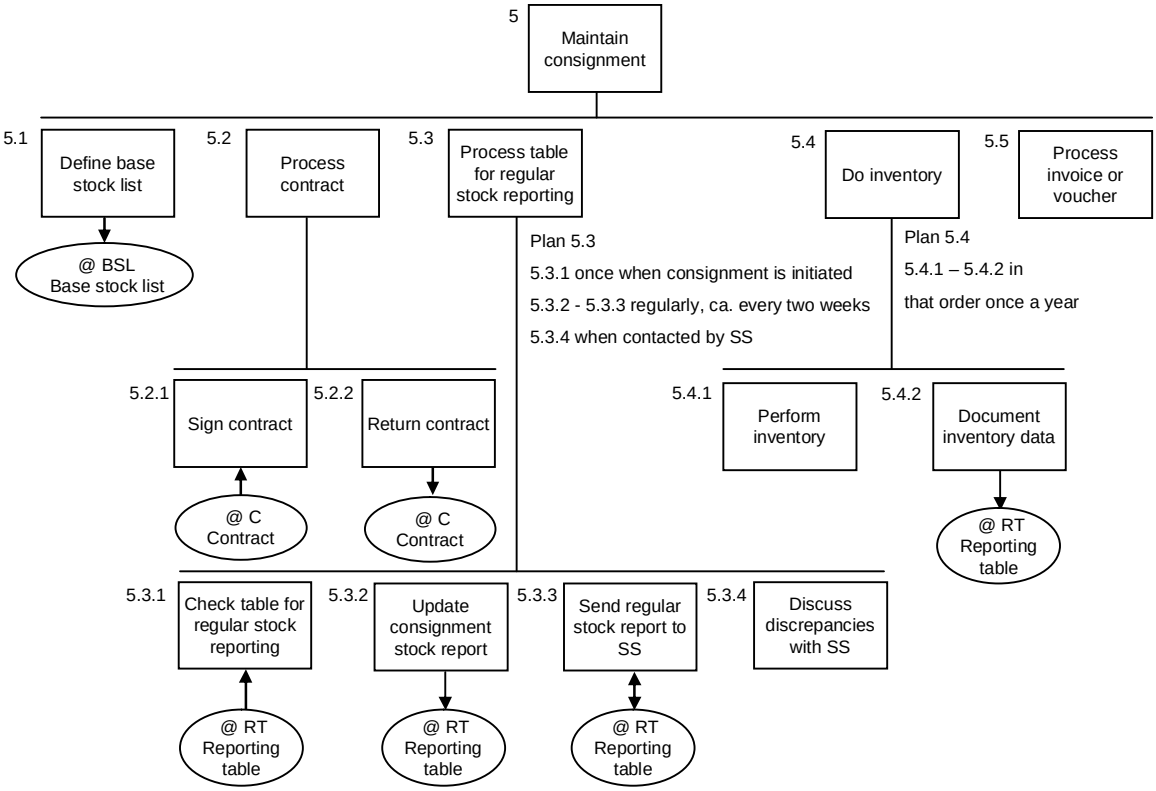


Figure B-6: Tasks of Customer (C)

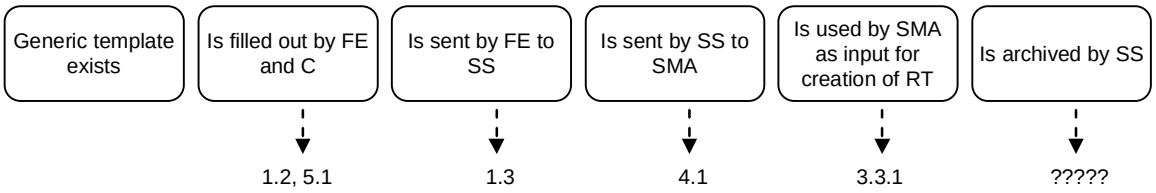


Figure B-7: Lifecycle of Base Stock List (BSL)

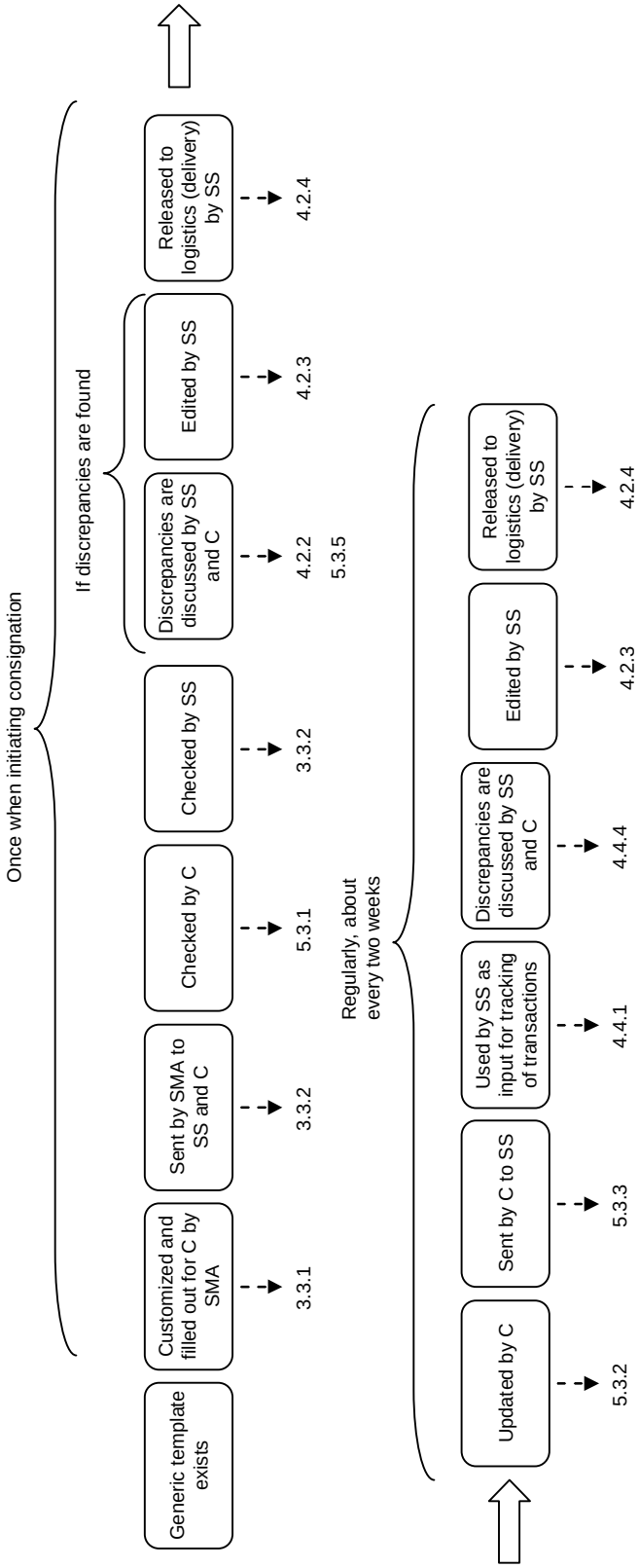


Figure B-8: Lifecycle of a Reporting Table (RT) (used by a customer for regular stock reporting of consignment items)

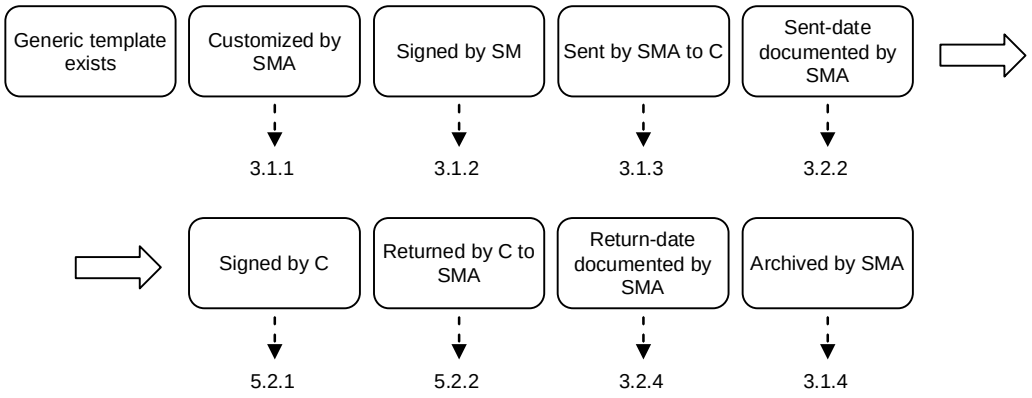


Figure B-9: Lifecycle of a Contract (C)

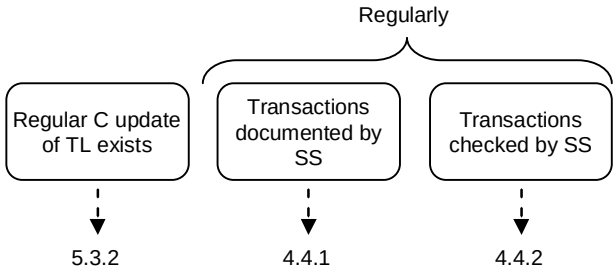


Figure B-10: Lifecycle of a Transaction Tracking List (TTL)

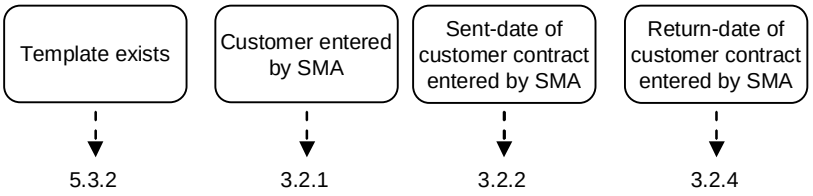


Figure B-11: Lifecycle of a Consignation Customers List (CCL)

Appendix C: Collaborative Task Handling

This appendix provides details about the collaborative task handling. It provides an overview of the processing of messages for exchange of tasks and deliverables from systems' perspective. The message attributes that are transferred as embedded meta-information are further provided.

C.1 Message Processing – Functional Flow

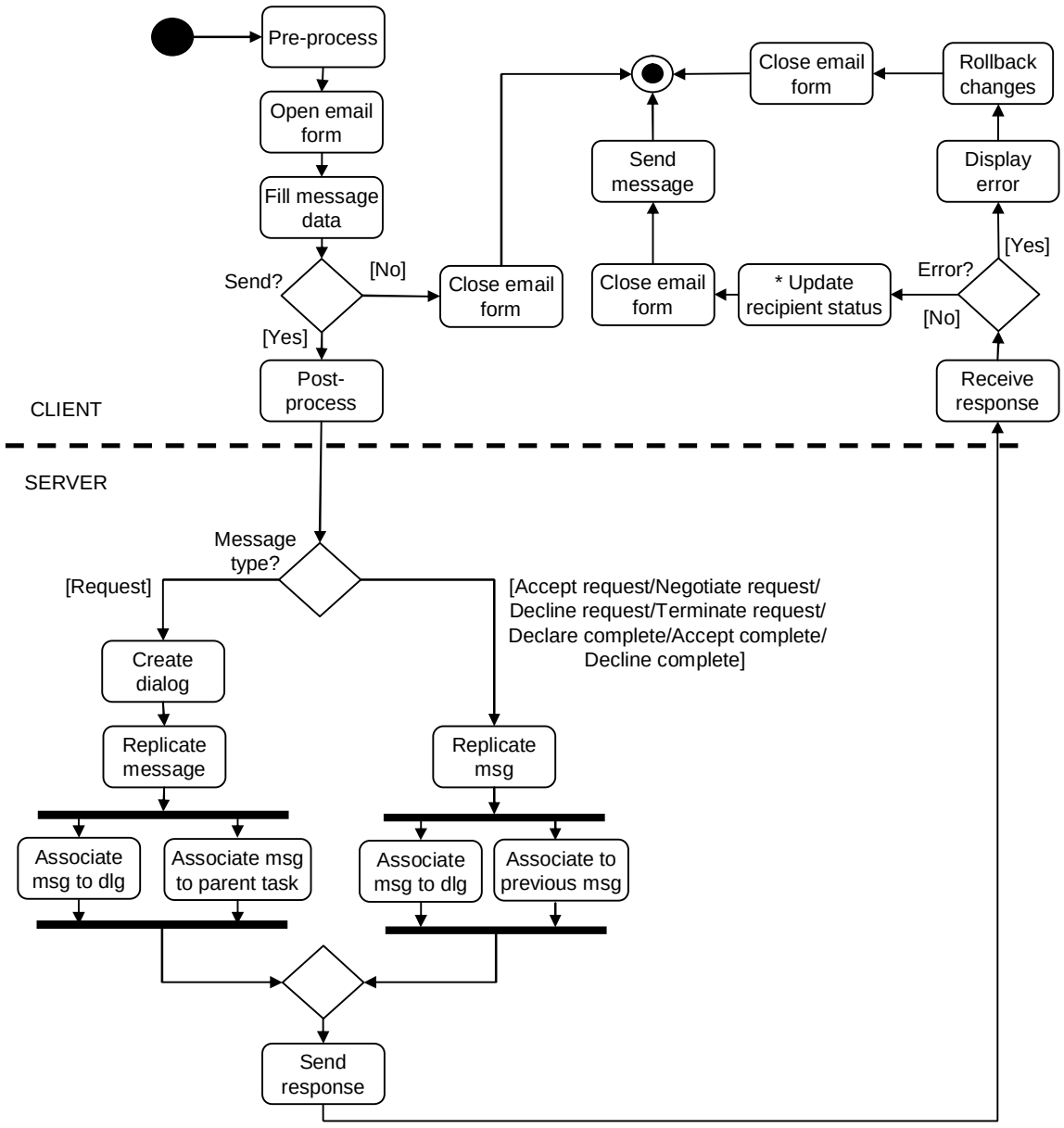


Figure C-1: Message processing - steps denoted with '*' are applicable only for messages sent by a task requester

C.2 Embedded Message Attributes

The message attributes, required to enable collaborative handling of task delegations and construction of task delegation graphs are shown in Table C-1 and described in the following:

- **identifier** - provides a unique identification of a message throughout the process composition environment to enable associations according to the task management model (cf. Figure 4.2).
- **type** - specifies the message type (request, request acceptance, request declination, request termination, completion declaration, acceptance or declination of completion declaration).
- **dialog identifier** - associates a message to a given dialog instance.
- **previous message identifier** - embedded in all messages except requests and request termination to enable their association to the dialog and correct ordering within the dialog.
- **root task identifier** - transmitted with requests, to enable association of the resulting accepted task at recipient site to the correct task delegation graph (process instance).
- **requester task identifier** - refers to the task, which was originally requested. This is the identifier of the *parent task* associated to a given request message according to the task management model (cf. Figure 4.2). This identifier is transmitted with requests, and inherited by the related response messages which the recipients send. When a task requester receives a response, this identifier is used to find the corresponding task in the requester's to-do list and to update its recipient status for the recipient that has responded. The requester task identifier is further embedded in request terminations. This identifier is the same in the request termination and the originally sent request, which allows mapping the termination as a response to the given request message.
- **recipient task identifier** - refers to the task, which emerged on recipient site from the acceptance of a request, i.e. this is the task that resulted from the agreement in a delegation dialog. According to the runtime task management model, this task has a *parent mail* association to the acceptance message (cf. Figure 4.2). The recipient task identifier is embedded in completion declarations that are sent by a task recipient, and inherited in related response messages of a task requester. When the task recipient receives the response for their completion declaration, this identifier is used to find the corresponding task in the recipient's to-do list and to update its status.
- **suggested task pattern identifier** - used to provide recommendation for further decomposition of delegated tasks based on a task pattern. This attribute enables unfolding end-to-end collaborative processes according to globally available task patterns.
- **task name** - transferred as message subject, followed by some identification of the message type. For example if a task with name "Maintain customer master data" is requested, the resulting request message receives a subject "Maintain customer master data / Request". This helps the users to directly identify what task is addressed and what kind of collaborative action is needed when looking at the message in their inbox.
- **task description** - embedded in request and negotiation messages, to precise the task being requested or negotiated. The description can be embedded as plain text. In email environments allowing some kind of formatting, e.g. through HTML message bodies, the task description may be separated from the request message text, e.g. through setting different background or input areas for both texts in a system-specific HTML form.
- **start date, due date** and **priority** - specify the task boundary conditions and are embedded in task requests and negotiations in human-readable form. These attributes can be embedded in textual form or with an enhanced visual representation if the email environment allows this, e.g. through HTML message bodies. Some environments like e.g. Microsoft Outlook support message prioritization, allowing direct transfer of task priority to message priority and vice versa.

Table C-1: Embedded message attributes for exchange of tasks and deliverables

		Message type							
		Request	Request negotiation	Request acceptance	Request declination	Request termination	Declare complete	Accept complete	Decline complete
Message attribute	identifier	x	x	x	x	x	x	x	x
	type	x	x	x	x	x	x	x	x
	dialog identifier		x	x	x		x	x	x
	previous message identifier		x	x	x		x	x	x
	root task identifier	x							
	requester task identifier	x	x	x	x	x	x		
	recipient task identifier						x	x	x
	suggested task pattern identifier	x	x						
	task name	x	x	x	x	x	x	x	x
	task description	x	x	x	x	x	x	x	x
	task start date	x	x						
	task due date	x	x						
	task priority	x	x						
Message send by	task requester	x	x			x		x	x
	task recipient		x	x	x		x		

Appendix D: TAM Evaluation Results

This appendix contains the results from the evaluation based on the Technology Acceptance Model (TAM) [Dav85, Dav89]. Descriptive and inferential statistics are provided.

D.1 Descriptive Statistics

This section provides tables with descriptive statistics on the TAM-based evaluation. In all tables **M** denotes the mean value, **SD** the standard deviation and **N** the number of answers on which the results are based. [Dav85, Dav89] group the questions into clusters focusing on different aspects of usefulness and ease of use. These clusters are indicated in the respective questions (**NA** stands for *not available*). The clusters are listed in the following according to [Dav85, Dav89]:

Clusters for usefulness:

- Cluster *A* relates to job effectiveness
- Cluster *B* relates to productivity and time savings
- Cluster *C* relates to importance of the system to the users' job
- Cluster *D* relates to control over the job

Clusters for ease of use are:

- Cluster *A* relates to physical effort
- Cluster *B* relates to mental effort
- Cluster *C* relates to how easy the system is to learn

Table D-1: Usefulness of CTM to-do list (N=13)

#	Cluster	Question	M	SD
1	A	Using the CTM to-do list would improve the quality of the work I do	1,38	0,77
2	D	Using the CTM to-do list would give me greater control over my work	1,38	1,26
3	B	The CTM to-do list would enable me to accomplish tasks more quickly	0,23	1,48
4	C	The CTM to-do list would support critical aspects of my job	0,92	1,19
5	B	Using the CTM to-do list would increase my productivity	0,23	1,48
6	A	Using the CTM to-do list would improve my job performance	0,54	1,51
7	B	Using the CTM to-do list would allow me to accomplish more work than would otherwise be possible	-0,38	1,12
8	A	Using the CTM to-do list would enhance my effectiveness on the job	0,62	1,33
9	C	Using the CTM to-do list would make it easier to do my job	0,69	1,44
10	NA	Overall, I would find the CTM to-do list useful in my job	1,08	1,38

Table D-2: Ease of use of the CTM to-do list (N=13)

#	Cluster	Question	M	SD
1	A	I find the CTM to-do list cumbersome to use.	-1,54	1,61
2	NA	Learning to operate the CTM to-do list is easy for me.	1,69	1,32
3	B	Interacting with the CTM to-do list is often frustrating.	-0,69	1,55
4	A	I find it easy to get the CTM to-do list to do what I want it to do.	1,46	0,88
5	A	The CTM to-do list is rigid and inflexible to interact with.	-1	1,15
6	C	It is easy for me to remember how to perform tasks using the CTM to-do list.	1,46	0,66
7	B	Interacting with the CTM to-do list requires a lot of mental effort.	-0,92	1,66
8	B	My interaction with the CTM to-do list is clear and understandable.	1,46	0,78
9	NA	I find it takes a lot of effort to become skillful at using the CTM to-do list	-1,69	0,85
10	NA	Overall, I find the CTM to-do list easy to use.	1,62	0,65

Table D-3: Usefulness of the CTM Task Delegation Graph (TDG) overview (N=13)

#	Cluster	Question	M	SD
1	A	Using the CTM TDG overview would improve the quality of the work I do	1,31	0,95
2	D	Using the CTM TDG overview would give me greater control over my work	1,69	0,95
3	B	The CTM TDG overview would enable me to accomplish tasks more quickly	0,77	1,17
4	C	The CTM TDG overview would support critical aspects of my job	1,31	0,85
5	B	Using the CTM TDG overview would increase my productivity	0,38	1,04
6	A	Using the CTM TDG overview would improve my job performance	0,31	1,03
7	B	Using the CTM TDG overview would allow me to accomplish more work than would otherwise be possible	-0,15	0,99
8	A	Using the CTM TDG overview would enhance my effectiveness on the job	0,62	1,19
9	C	Using the CTM TDG overview would make it easier to do my job	0,62	0,96
10	NA	Overall, I would find the CTM TDG overview useful in my job	1,15	1,07

Table D-4: Ease of use of the CTM Task Delegation Graph (TDG) overview (N=13)

#	Cluster	Question	M	SD
1	A	I find the CTM TDG overview cumbersome to use.	-2,31	0,75
2	NA	Learning to operate the CTM TDG overview is easy for me.	1,92	0,64
3	B	Interacting with the CTM TDG overview is often frustrating.	-1,54	1,05
4	A	I find it easy to get the CTM TDG overview to do what I want it to do.	1,38	1,12
5	A	The CTM TDG overview is rigid and inflexible to interact with.	-1,23	1,24
6	C	It is easy for me to remember how to perform tasks using the CTM TDG overview.	1,69	0,95
7	B	Interacting with the CTM TDG overview requires a lot of mental effort.	-1,77	1,01
8	B	My interaction with the CTM TDG overview is clear and understandable.	2	0,71
9	NA	I find it takes a lot of effort to become skillful at using the CTM TDG overview	-1,62	1,39
10	NA	Overall, I find the CTM TDG overview easy to use.	1,92	0,86

Table D-5: Usefulness of the CTM Task Delegation Dialog (TDD) overview (N=13)

#	Cluster	Question	M	SD
1	A	Using the CTM TDD overview would improve the quality of the work I do	1,15	0,69
2	D	Using the CTM TDD overview would give me greater control over my work	1,31	0,48
3	B	The CTM TDD overview would enable me to accomplish tasks more quickly	0,77	0,60
4	C	The CTM TDD overview would support critical aspects of my job	1,08	0,95
5	B	Using the CTM TDD overview would increase my productivity	0,46	0,78
6	A	Using the CTM TDD overview would improve my job performance	0,46	0,66
7	B	Using the CTM TDD overview would allow me to accomplish more work than would otherwise be possible	0,08	0,64
8	A	Using the CTM TDD overview would enhance my effectiveness on the job	0,69	0,75
9	C	Using the CTM TDD overview would make it easier to do my job	0,77	0,73
10	NA	Overall, I would find the CTM TDD overview useful in my job	1,31	0,48

Table D-6: Ease of use of the CTM Task Delegation Dialog (TDD) overview (N=13)

#	Cluster	Question	M	SD
1	A	I find the CTM TDD overview cumbersome to use.	-1,38	1,33
2	NA	Learning to operate the CTM TDD overview is easy for me.	1,54	0,97
3	B	Interacting with the CTM TDD overview is often frustrating.	-1,31	1,25
4	A	I find it easy to get the CTM TDD overview to do what I want it to do.	1,08	0,86
5	A	The CTM TDD overview is rigid and inflexible to interact with.	-1,08	1,26
6	C	It is easy for me to remember how to perform tasks using the CTM TDD overview.	1,46	0,88
7	B	Interacting with the CTM TDD overview requires a lot of mental effort.	-1,15	1,34
8	B	My interaction with the CTM TDD overview is clear and understandable.	1,23	1,01
9	NA	I find it takes a lot of effort to become skillful at using the CTM TDD overview	-1	1,58
10	NA	Overall, I find the CTM TDD overview easy to use.	1,23	1,01

Table D-7: Usefulness of the CTM task patterns (N=13)

#	Cluster	Question	M	SD
1	A	Using the CTM task patterns would improve the quality of the work I do	0,92	1,75
2	D	Using the CTM task patterns would give me greater control over my work	0,77	1,83
3	B	The CTM task patterns would enable me to accomplish tasks more quickly	0,62	1,66
4	C	The CTM task patterns would support critical aspects of my job	0,69	1,65
5	B	Using the CTM task patterns would increase my productivity	0,54	1,90
6	A	Using the CTM task patterns would improve my job performance	0,62	1,80
7	B	Using the CTM task patterns would allow me to accomplish more work than would otherwise be possible	0,31	1,60
8	A	Using the CTM task patterns would enhance my effectiveness on the job	0,69	1,84
9	C	Using the CTM task patterns would make it easier to do my job	0,77	1,92
10	NA	Overall, I would find the CTM task patterns useful in my job	0,77	1,79

Table D-8: Ease of use of the CTM Task Pattern Explorer (TPE) (N=5)

#	Cluster	Question	M	SD
1	A	I find the CTM TPE cumbersome to use.	-1	0,71
2	NA	Learning to operate the CTM TPE is easy for me.	0,8	0,45
3	B	Interacting with the CTM TPE is often frustrating.	0	1,22
4	A	I find it easy to get the CTM TPE to do what I want it to do.	0,6	0,55
5	A	The CTM TPE is rigid and inflexible to interact with.	-1	0,71
6	C	It is easy for me to remember how to perform tasks using the CTM TPE.	1,2	0,45
7	B	Interacting with the CTM TPE requires a lot of mental effort.	-1	0,71
8	B	My interaction with the CTM TPE is clear and understandable.	1	0,71
9	NA	I find it takes a lot of effort to become skillful at using the CTM TPE.	-0,8	0,84
10	NA	Overall, I find the CTM TPE easy to use.	1,2	0,45

Table D-9: Usefulness of the CTM Task Evolution Explorer (TEE) (N=13)

#	Cluster	Question	M	SD
1	A	Using the CTM TEE would improve the quality of the work I do.	0,15	1,86
2	D	Using the CTM TEE would give me greater control over my work	0,46	1,90
3	B	The CTM TEE would enable me to accomplish tasks more quickly	-0,46	1,39
4	C	The CTM TEE would support critical aspects of my job	0,08	1,75
5	B	Using the CTM TEE would increase my productivity	-0,54	1,45
6	A	Using the CTM TEE would improve my job performance	-0,54	1,45
7	B	Using the CTM TEE would allow me to accomplish more work than would otherwise be possible	-0,62	1,56
8	A	Using the CTM TEE would enhance my effectiveness on the job	-0,31	1,49
9	C	Using the CTM TEE would make it easier to do my job	0	1,73
10	NA	Overall, I would find the CTM TEE useful in my job	0,08	1,85

Table D-10: Usefulness of the CTM Process Transformation (PT) (N=13)

#	Cluster	Question	M	SD
1	A	Using the CTM PT would improve the quality of the work I do.	0,46	1,61
2	D	Using the CTM PT would give me greater control over my work.	0,92	1,85
3	B	The CTM PT would enable me to accomplish tasks more quickly.	0,15	1,57
4	C	The CTM PT would support critical aspects of my job.	0,38	1,66
5	B	Using the CTM PT would increase my productivity.	-0,15	1,46
6	A	Using the CTM PT would improve my job performance.	0	1,47
7	B	Using the CTM PT would allow me to accomplish more work than would otherwise be possible.	-0,38	1,26
8	A	Using the CTM PT would enhance my effectiveness on the job.	-0,23	1,54
9	C	Using the CTM PT would make it easier to do my job.	0,23	1,54
10	NA	Overall, I would find the CTM PT useful in my job.	0,62	1,71

D.2 Inferential Statistics

This section provides inferential statistics. Two-sample, non-parametric tests focus on differences between TAM metrics of different user groups. Various grouping criteria are used, which are indicated in the corresponding result tables.

Correlation test results are further provided, focusing on the relationships between TAM usefulness metrics and number of managed persons, and between usefulness metrics for different concepts.

Table D-11: Results from a Mann-Whitney U test for overall **usefulness** estimations of users involved in short-term (N=7) and long-term (N=6) CTM usage (grouping variable *usage = 0* or *1*)

	Overall usefulness of:					
	to-do list	task delegation graph	task delegation dialog	task patterns	task evolution tracing	workflow generation
Mann-Whitney U	19,500	5,000	15,500	16,000	20,500	19,000
Wilcoxon W	47,500	26,000	36,500	37,000	41,500	47,000
Z	-,233	-2,570	-,980	-,783	-,078	-,306
Asymp. Sig. (2-tailed)	,816	,010	,327	,434	,938	,760
Exact Sig. [2*(1-tailed Sig.)]	,836[a]	**,022[a]**	,445[a]	,534[a]	,945[a]	,836[a]

a. Not corrected for ties.

Table D-12: Results from a Mann-Whitney U test for overall **ease of use** estimations of users involved in short-term (N=7) and long-term (N=6) CTM usage (grouping variable *usage = 0* or *1*)

	Overall ease of use of:		
	to-do list	task delegation graph	task delegation dialog
Mann-Whitney U	18,000	15,500	20,000
Wilcoxon W	39,000	36,500	48,000
Z	-,477	-,861	-,153
Asymp. Sig. (2-tailed)	,633	,389	,879
Exact Sig. [2*(1-tailed Sig.)]	,731[a]	,445[a]	,945[a]

a. Not corrected for ties.

Table D-13: Spearman correlation test for overall **usefulness** estimations and number of managed persons: r_s is the correlation coefficient, p is significance, and N is number of test items (gray cells are repetitions and can be ignored)

			Num. of managed persons	Overall usefulness of:					
				to-do list	task delegation graph	task delegation dialog	task patterns	task evolution tracing	workflow generation
Number of managed persons		r_s	1,000	-,491	-,423	,000	-,594*	-,361	-,142
		p (2-tailed)	.	,088	,150	1,000	,032	,226	,643
		N	13	13	13	13	13	13	13
Overall usefulness of:	to-do list	r_s	-,491	1,000	,007	,218	,368	,231	-,158
		p (2-tailed)	,088	.	,983	,475	,216	,447	,606
		N	13	13	13	13	13	13	13
	task delegation graph	r_s	-,423	,007	1,000	,300	,433	,088	,106
		p (2-tailed)	,150	,983	.	,319	,139	,776	,731
		N	13	13	13	13	13	13	13
	task delegation dialog	r_s	,000	,218	,300	1,000	,171	-,438	-,095
		p (2-tailed)	1,000	,475	,319	.	,577	,134	,757
		N	13	13	13	13	13	13	13
	task patterns	r_s	**-,594***	,368	,433	,171	1,000	,408	,200
		p (2-tailed)	**,032**	,216	,139	,577	.	,166	,513
		N	13	13	13	13	13	13	13
	task evolution	r_s	-,361	,231	,088	-,438	,408	1,000	,604*
		p (2-tailed)	,226	,447	,776	,134	,166	.	,029
		N	13	13	13	13	13	13	13
	workflow generation	r_s	-,142	-,158	,106	-,095	,200	**,604***	1,000
		p (2-tailed)	,643	,606	,731	,757	,513	**,029**	.
		N	13	13	13	13	13	13	13

*. Correlation is significant at the 0.05 level (2-tailed)

Curriculum Vitae

September 04, 1980	Born in Sofia, Bulgaria
09/1987 – 06/1994	67 Primary School “Vasil Drumev“ Sofia, Bulgaria
09/1994 – 06/1999	91 German Language High School “Prof. Konstantin Galabov” Sofia, Bulgaria Graduated with a high school diploma
09/1999 – 04/2005	Technical University Sofia, Bulgaria, Faculty of German Engineering and Business Management Education Graduated with a Master degree in engineering from the Technical University Sofia, Bulgaria and Diplom-Ingenieur degree from the Fridericiana University Karlsruhe, Germany
05/2001 – 09/2005	Software Engineer in the area of B2B integration at the subsidiary of Seeburger AG in Sofia, Bulgaria
10/2005 – 08/2006	Software Engineer at the SAP NetWeaver kernel team in SAP Labs Bulgaria
09/2006 – 06/2009	Ph.D. Student at the SAP Research Campus Based Engineering Center Darmstadt, Germany Graduated with a doctoral degree in informatics from the Technical University Darmstadt